Guide to VAX/VMS System Management and Daily Operations

Order Number: AA-Y507A-TE

September 1984

This manual is a conceptual and tutorial guide for all users responsible for managing and operating a VAX/VMS system.

Revision/Update Information: This is a new manual.

Software Version: VAX/VMS Version 4.0

digital equipment corporation
maynard, massachusetts

September 1984

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

Contents

PART II MAINTAINING THE SYSTEM

FIGURES

TABLES

Preface

The *Guide to VAX/VMS System Management and Daily Operations* is a conceptual and tutorial guide to system management and daily operations on one or more VAX/VMS systems. This manual is designed to

- Describe the tasks involved in managing, operating, and maintaining a VAX/VMS system
- Provide general information on day-to-day operating procedures
- Introduce utilities and commands used to perform specific system management functions

The manual is not a one-volume reference source of information. Utilities and commands used to perform specific system management and operating tasks are described in detail in separate sections of both the *VAX/VMS Utilities Reference Volume* and the *VAX/VMS DCL Dictionary*. The manual does not include information on configuring and managing VAXcluster systems; for that information, refer to the *Guide to VAXclusters*.

Intended Audience

This manual addresses experienced users of the VAX/VMS system who must perform the functions of a system manager or operator.

Structure of This Document

The *Guide to VAX/VMS System Management and Daily Operations* is divided into two parts, each covering a related group of system management tasks:

- Part I, *Setting Up the System*, describes the tasks and tools necessary to prepare a VAX/VMS operating system for daily operation. This section includes discussions on making site-specific modifications to VAX/VMS, performing system startup and shutdown, setting up and maintaining user accounts, and allocating system resources.

- Part II, *Maintaining the System,* describes the tasks and tools necessary to maintain and routinely operate the system, and builds upon concepts presented in Part I. This section includes discussions on maintaining public files and volumes, installing images as known images, maintaining batch and output queues, using error and operator logging facilities, and modifying system parameters.

Appendix A describes special operations for the VAX–11 /750 processor. Appendix B provides information about the Computer Interconnect (CI), associated error messages, and corrective procedures.

Associated Documents

For general background information about the system, see the *Introduction to VAX/VMS.*

For information on system installation and upgrade procedures, consult the *Guide to VAX/VMS Software Installation* and the software installation booklets for your VAX processor.

For hardware operating instructions, refer to the appropriate hardware manual.

For managing network operations, refer to the *Guide to Networking on VAX/VMS.*

For configuring and managing VAXclusters, refer to the *Guide to VAXclusters.*

The following documents provide important system management reference and supplemental information:

- *VAX/VMS DCL Dictionary*
- *VAX/VMS Utilities Reference Volume*
- *VAX/VMS System Messages and Recovery Procedures Reference Manual*
- *Guide to VAX/VMS Disk and Magnetic Tape Operations*
- *Guide to VAX/VMS Performance Management*

Conventions Used in This Document

Convention	Meaning
RET	A symbol with a one- to six-character abbreviation indicates that you press a key on the terminal, for example, RET .
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O.
$ SHOW TIME 05-JUN-1985 11:55:22	Command examples show in black letters all output lines or prompting characters that the system prints or displays. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, means either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

1 Introduction

A computer installation exists to provide its users with efficient and economical service. The challenge of system management is to provide such service on a day-to-day basis.

Management of a VAX/VMS installation entails two main responsibilities: system performance and system operation. These responsibilities require that you

- Make decisions that relate to optimizing the overall performance and efficiency of the system
- Perform tasks that relate to the overall management and control of the system

This manual discusses the issues and facts that you must understand to make decisions, and it provides guidelines for performing system management functions.

It is not possible to prescribe a set of formulas for setting up and running a VAX/VMS system, because you cannot manage a system by rote. Rather, you must—by combining an understanding of users' needs with a working knowledge of system capabilities—work out your own strategy.

System management tasks can be performed by a single individual, or by a system manager assisted by one or more operators. Since responsibilities vary from site to site, this manual does not make job distinctions between system managers and operators. As you read the manual, therefore, it may appear at times that one person is performing all system management functions. Bear in mind that this is not always the case.

1.1 System Management Tasks

System management tasks typically fall into the following categories:

- Installing and upgrading the system
- Making site-specific modifications
- Controlling system operation
- Configuring the system for good performance
- Planning to meet future requirements

The first task is the subject of the *Guide to VAX/VMS Software Installation* and the installation booklets for your VAX processor. The next three are described in this manual. The final task is beyond the scope of this manual.

As system manager, you can perform many of your duties from user terminals. However, you must perform one of your chief functions, communicating with other users, at an *operator's terminal*, which you define by using the privileged DCL command REPLY/ENABLE (see the *VAX/VMS DCL Dictionary*).

Your relationship with users is largely based on messages that pass between you and them. A record of these messages is displayed on the operator's terminal; the messages are also entered in the operator's log file for later reference.

You can perform certain system management functions only from the system console terminal. These functions include bootstrapping the system and communicating with the VAX processor's console subsystem.

1.1.1 Operational Tasks

The VAX/VMS system normally runs with minimal operator intervention. In many installations, however, operators keep the system running smoothly by performing some of the following tasks:

- Physically mounting magnetic tapes and disks at the request of the users who own them
- Initializing and mounting system volumes
- Backing up public files and volumes
- Sending messages to users
- Tending line printers and card readers
- Responding to emergencies
- Printing copies of the operator's log file and the error log file
- Shutting down and restarting the system
- Bringing up and shutting down network components

To carry out these tasks effectively, operators usually consult the system manager for guidance on site-specific policies and procedures.

1.1.2 Tools and Privileges

To manage a VAX/VMS system, you have at your disposal powerful commands and utilities to monitor and control system operations and resources. In addition, you receive the privileges necessary to carry out your management functions.

1.1.2.1 DIGITAL Command Language Commands

This manual contains numerous references to the DIGITAL Command Language (DCL) commands used most often to keep a VAX/VMS system running smoothly. Most of these commands require the OPER user privilege. For detailed descriptions of these commands, see the *VAX/VMS DCL Dictionary*.

1.1.2.2 VAX/VMS Utilities

Table 1–1 lists those utilities frequently employed primarily as system management tools and gives a brief description of their purpose. You can find further details on each of these utilities, as well as other utilities with which you will want to become familiar, in the *VAX/VMS Utilities Reference Volume.*

Table 1–1 System Management Utilities

Utility Name	Function
ACCOUNTING	Produces reports and summaries of system usage
ANALYZE/ERROR_LOG	Reports the contents of the system error log file
AUTHORIZE	Adds and modifies records in the existing user authorization and network authorization files or creates new files; adds and modifies records in the rights database
BAD	Analyzes block-addressable devices and records the location of blocks that cannot reliably store data
DISKQUOTA	Controls the usage of disk volumes
INSTALL	Installs and maintains known images
MONITOR	Monitors system-wide performance
SYSGEN	Performs tasks associated with system generation such as loading and connecting drivers, creating or extending swapping and paging files, displaying or modifying the values of the system parameters, and enabling multiport memory units

1.1.2.3 Privileges

System management functions require privileges that are denied to most users. Chapter 6 provides more detailed information about the privileges and who should receive them. Chapter 5 describes how to set up authorization records that grant these privileges.

1.1.3 VAX/VMS Operating System Components

To manage a VAX/VMS system effectively, you must be familiar with its principal components, which are contained in ten directories on the system distribution medium. The logical names of these directories and brief descriptions of their contents are as follows:

- SYS$ERRORLOG—Contains the error log file (ERRLOG.SYS)
- SYS$EXAMPLES—Contains sample driver programs, user-written system services, and other source programs
- SYS$HELP—Contains text files and help libraries for the Help Utility
- SYS$LIBRARY or SYS$SHARE—Contains various macro and object libraries and shareable images
- SYS$MAINTENANCE—Contains system diagnostic programs
- SYS$MANAGER—Contains files used in managing the operating system
- SYS$MESSAGE—Contains system message files
- SYS$SYSTEM—Contains the executable images of most operating system functions
- SYS$TEST—Contains files used in testing operating system functions
- SYS$UPDATE—Contains files used in applying system updates

For a complete list of the files contained on the distribution medium, see the *Guide to VAX/VMS Software Installation.*

PART I Setting Up the System

2 Customizing the Operating System After Installation

Your first responsibility as system manager is to get your VAX/VMS system up and running. At a minimum, you must install and bootstrap the VAX/VMS operating system supplied by DIGITAL, following procedures described in the installation documentation for your VAX processor. Refer to this documentation for information on all installation tasks.

Once your system is installed, you can customize it for your particular operating environment by performing one or more of the following operations:

- **Build a site-specific startup command procedure.**

  When you install your system, the site-specific startup command procedure, SYS$MANAGER:SYSTARTUP.COM, is blank to accommodate the installation. After you have installed the system, you add commands to SYSTARTUP.COM to set up an environment specific to your site. For example, you normally include commands that initialize and start batch and output queues. Section 2.2 describes startup command procedures; Section 2.3 shows how to create a typical site-specific startup command procedure.

- **Select a default bootstrap command procedure.**

  The file DEFBOO.CMD is the bootstrap command procedure invoked by default to bootstrap your system from the appropriate disk drive, as explained in Section 2.4. After installation, you must select and copy to DEFBOO.CMD a default bootstrap command procedure from those available on your console medium. You can select either a *nonstop* or a *conversational* procedure. You select a nonstop procedure if you want to bootstrap the system without stopping to change system parameters. A conversational procedure lets you change parameters in the course of the boot. Sections 2.4 and 2.5 explain how to select, and if necessary, modify the procedure appropriate for your site. Appendix A discusses bootstrap operations peculiar to VAX–11/750 systems.

- **Make site-specific modifications for special configuration or workload needs.**

  When you bootstrap your system during installation, the AUTOGEN command procedure generates system parameters that are suitable for your hardware. However, you may want to modify some of them if you have an unusual hardware configuration or special workload requirements. In that case, refer to the discussion of AUTOGEN in Chapter 11.

  If you have a dual-RL02 VAX–11/730 system, you will need to adjust the sizes of the paging and swapping files to accommodate crash dumps. Additionally, if your VAX–11/730 system uses a DMF32 "combo" board, you must redefine the line printer device names. These modifications are described in Sections 2.6.1 and 2.6.2.

  Once you have made the necessary site-specific modifications to the operating system, you must shut the system down and reboot it to ensure that they are installed. For example, most modifications to system parameters do not take effect until the system is shut down and rebooted. The procedures for shutting down and rebooting the system are explained in Section 2.7 and in Chapter 4.

- **Back up the system.**

  After installing and customizing your system, you must perform various backup operations as described in Section 2.8.

The remainder of this chapter discusses the tasks that you as system manager perform to customize your newly installed VAX/VMS system and prepare it for daily use.

Chapter 3 describes procedures for tailoring small-disk systems.

2.1 Logging In to the System

Before you can make site-specific modifications, you must log in to the system manager's account, SYSTEM. When you bootstrap the system, it announces itself with the initialization message:

```
  VAX/VMS Version 4.0 <dd-mmm-yyyy hh:mm:ss.s>

%%%%%%%%%%%  OPCOM, <dd-mmm-yyyy hh:mm:ss.s>  %%%%%%%%%%%
Logfile has been initialized by operator _OPA0:
Logfile is SYS$SYSROOT:[SYSMGR]OPERATOR.LOG;1

%SET-I-INTSET, login interactive limit = 64, Current interactive value = 0
  SYSTEM        job terminated at <dd-mmm-yyyy hh:mm:ss.s>
```

Use the following procedure to log in to the system as system manager:

1. Press the RETURN key on the console terminal.
2. In response to the system's request for your *username,* type SYSTEM.
3. In response to the system's request for your *password,* type the SYSTEM password.

Note: **DIGITAL recommends that you change the system manager account password frequently to maintain system security. Also, the account has full privileges by default, so you must exercise caution when using it. Chapter 6 provides additional information about the SYSTEM account.**

When you enter your password, the system prints a welcome message on the console terminal, followed by the time of of your last login:

```
Welcome to VAX/VMS Version 4.0
Last interactive login at 15-APR-1984 15:13:21.07
```

When the console terminal prints the command prompt, you can begin modifying the system using the instructions in the following sections.

2.2 Startup Command Procedures

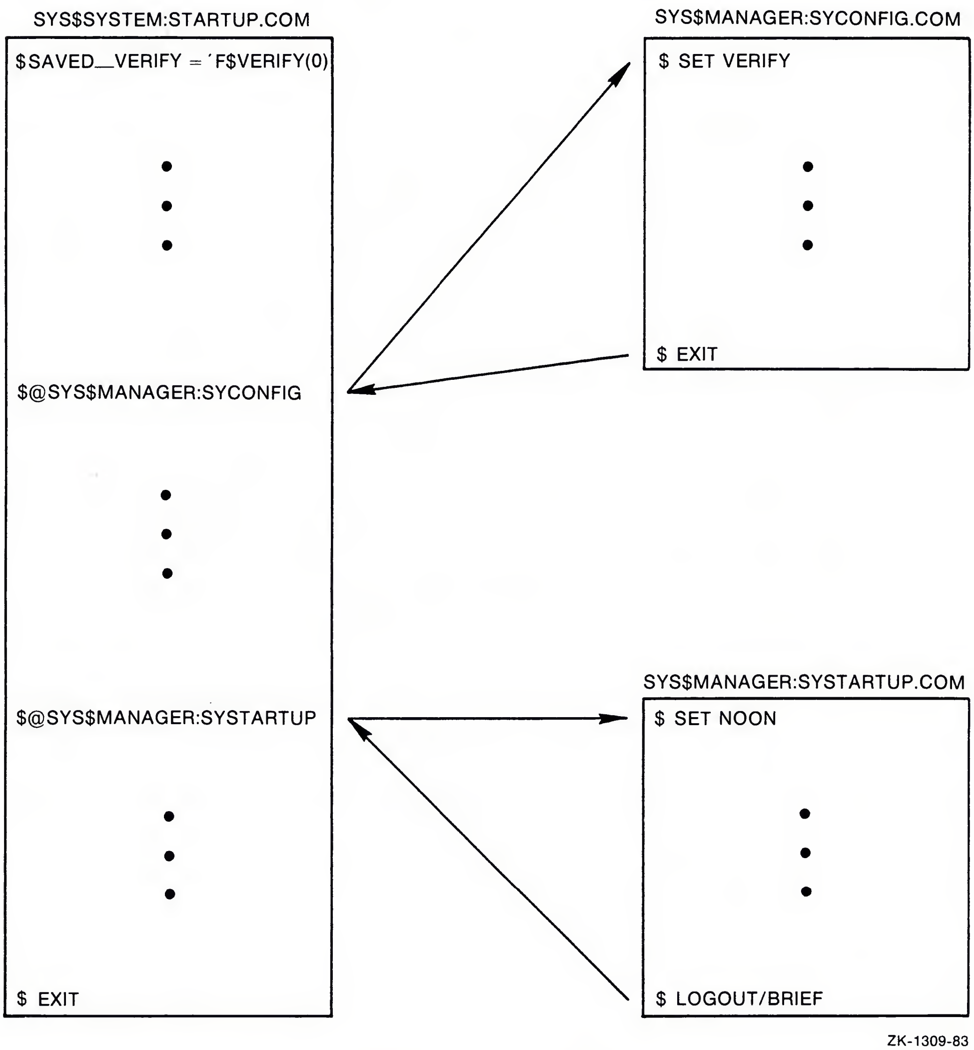
The software distribution kit includes the following startup command procedures:

- **SYS$SYSTEM:STARTUP.COM**—a file containing commands that must execute at initialization time in order for the system to run properly. This procedure is the site-independent startup command procedure supplied by DIGITAL.
- **SYS$MANAGER:SYCONFIG.COM**—an empty file supplied by DIGITAL to which you can add your own specific device configuration commands. This command procedure is invoked by STARTUP.COM (see Section 2.2.2).
- **SYS$MANAGER:SYSTARTUP.COM**—an empty file supplied by DIGITAL to which you can add specific initialization commands for your site. This procedure is the site-specific startup command procedure invoked by STARTUP.COM (see Section 2.2.3).

Caution: **Do not modify SYS$SYSTEM:STARTUP.COM. Instead, put site-specific commands in SYSTARTUP.COM, because new versions of VAX/VMS delete and replace the STARTUP.COM command file. By modifying SYSTARTUP.COM, you ensure that commands specific to your site are not lost whenever you upgrade your system to a new version of VAX/VMS. Also, leaving STARTUP.COM intact prevents you from inadvertently altering any commands in the file.**

Figure 2–1 illustrates the execution sequence of the startup command procedures.

Figure 2–1 Execution Sequence of Startup Command Procedures



2.2.1 Site-Independent Startup Command Procedure

The command procedure SYS$SYSTEM:STARTUP.COM is automatically executed immediately after the operating system is bootstrapped and includes commands that

- Perform various housekeeping chores
- Define system-wide logical names required for the symbolic debugger, language processors, linker, image activator, and help processor
- Start three processes that control error logging, the job controller, and the operator's log
- Connect devices that are physically attached to the system and load their I/O drivers (see Section 2.2.2)
- Install known images

Before STARTUP.COM terminates, it invokes the site-specific startup command procedure, SYS$MANAGER:SYSTARTUP.COM (see Section 2.2.3). When SYSTARTUP.COM completes, control is returned to STARTUP.COM and the number of interactive users is set to 64 by default, unless the number was set in the SYSTARTUP procedure (see Section 2.3.12). After this step, STARTUP.COM logs off.

2.2.2 Device Configuration Command Procedure

One step in the site-independent startup command procedure SYS$SYSTEM:STARTUP.COM is to connect the devices that are physically attached to the system and load their I/O drivers. To do this, STARTUP.COM first invokes the command procedure SYS$MANAGER:SYCONFIG.COM. (Remember that this command procedure comes with the software distribution kit as an empty file.) You can leave the file empty, or you can add your own specific device configuration commands.

After SYCONFIG.COM executes, control is returned to STARTUP.COM and the following commands are executed to connect all devices automatically and load their drivers:

```
$ RUN SYS$SYSTEM:SYSGEN
$ AUTOCONFIGURE ALL
```

You can suppress this autoconfiguration by including the following command as the last line in SYCONFIG.COM:

```
$ STARTUP$AUTOCONFIGURE_ALL == 0
```

2.2.3 Site-Specific Startup Command Procedure

The command procedure SYS$MANAGER:SYSTARTUP.COM is invoked by STARTUP.COM to execute startup commands that are specific to your site. DIGITAL recommends that you edit SYSTARTUP.COM to include commands that will perform the following typical site-specific operations:

- Mount public disks
- Assign logical names
- Set the characteristics of terminals and other devices
- Initialize and start queues
- Install known images
- Run the System Dump Analyzer
- Purge the operator's log file
- Submit batch jobs that are run at the time the system is initialized and that are periodically resubmitted
- Connect devices and multiport memory units
- Install secondary paging and swapping files
- Announce that the system is up and running
- Redefine the number of interactive users
- Terminate the command procedure

To minimize processing overhead when executing SYSTARTUP.COM, you normally include the DCL command SET NOON at the beginning of the file to disable error processing.

2.3 Creating Your Site-Specific Startup Command Procedure

The following sections describe how to create a typical site-specific startup command procedure. Any commands shown are provided as models; do not copy them line for line.

2.3.1 Mounting Public Disks

You will probably want to include mount commands to mount your public disks for system-wide access. You should also consider some of the advantages of using logical volume names to conceal the physical devices. When you mount a disk, MOUNT produces a special logical name called a logical volume name that you can use to reference the volume. When you dismount the volume, the logical name is deleted. For example:

```
$ MOUNT/SYSTEM  DRA1:  USERFILES  USER
```

This command produces the logical volume name USER and equates it to the concealed device name string DRA1:. However, USER only translates to a physical device while the data disk is actually mounted.

If you mount a disk and do not give an explicit logical volume name, MOUNT assigns a default name of the form: DISK$volume_label. In the example above, if no logical volume name were specified, the default logical volume name would have been DISK$USERFILES. Since the logical volume name is printed on the flag page of listings, and displayed on the terminal by the DCL commands SHOW DEVICE/FILES and SHOW MEMORY/FILES, you may occasionally see such labels.

If you and the users consistently use the logical volume name, it is not necessary to know on which physical drive the volume is mounted. Thus, you can avoid including physical device names in programs and command procedures.

Note that when you run SYSTARTUP.COM (and only then), the default on the MOUNT command is /NOASSIST. This qualifier means that operator-assisted mounts are disabled. If you want to enable this feature during SYSTARTUP, you must specify /ASSIST with each MOUNT command. See Chapter 7 for more information on mounting public disks.

2.3.2 Assigning System-Wide Logical Names

In addition to the logical names assigned in the site-independent startup command procedure, you can assign system-wide logical names with a command like the following:

```
$ DEFINE/SYSTEM SYSDSK SYS$SYSDEUCE:
```

See the *VAX/VMS DCL Dictionary* for more detailed information on logical name assignments.

2.3.3 Setting Device Characteristics

To establish the characteristics of the terminals and other devices on the system, you can use a series of SET commands in a command procedure. You may want to include comments that give the user names for terminal owners. For example:

```
$ SET TERMINAL TTC2: /SPEED=300/DEVICE_TYPE=LA36/PERMANENT  !JONES
$ SET TERMINAL TTD1: /SPEED=9600/PERMANENT                  !WRENS
$ SET TERMINAL TTD4: /SPEED=1200/PERMANENT                  !JRSMITH
$ SET TERMINAL TTG4: /SPEED=1200/MODEM/PERMANENT            !DIALUP1
$ SET PRINTER LPA0: /LOWER/NOCR
$ SET DEVICE LPA0: /SPOOLED
```

Note that the /SPEED qualifier sets both transmission and reception speeds to the same value. The /MODEM qualifier defines a terminal for use on a dial-in line. Printer characteristics (SET PRINTER and SET DEVICE above) must be set prior to establishing queues for the printers. You may want to include SET TERMINAL commands in a separate file, named for example, TERMSET.COM, and include a command in SYSTARTUP.COM to invoke the TERMSET.COM command file. When the command file finishes execution, control returns to SYSTARTUP.COM.

2.3.4 Initializing and Starting Queues

Include queue commands to start the system job queue manager, establish spooled devices, and set up print and batch queues.

Initialize and start each queue with a separate INITIALIZE /QUEUE/START command line. The following commands start the system job queue manager and initialize and start all queues if they are stopped.

```
$ !
$ !Start the system job queue manager
$ !
$ START/QUEUE/MANAGER
$ !
$ !Set printers spooled and establish printer queues
$ !
$ SET PRINTER/LOWER  LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/NOENABLE_GENERIC  LPA0:
$ !
$ SET PRINTER/LOWER  LPB0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPB0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/NOENABLE_GENERIC  LPB0:
$ !
$ INITIALIZE/QUEUE/START/GENERIC=(LPA0,LPB0) SYS$PRINT
$ !
$ !Establish batch queues
$ !
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=2/BASE_PRIORITY=3 SYS$BATCH
```

On systems with a large number of queues, you may want to include queue commands in a separate file, named for example, STARTQ.COM, and include a command in SYSTARTUP.COM to invoke the queue command file. When the queue command file finishes execution, control returns to SYSTARTUP.COM.

Chapter 9 describes how to set up queues and establish spooled devices.

2.3.5 Installing Known Images

It is important to install user and system programs (in addition to the ones installed in the site-independent startup command procedure) so that they can be located quickly, shared, or provided privileges:

```
$ INSTALL = "INSTALL/COMMAND_MODE"
$ INSTALL
ADD/OPEN/SHARED/HEADER_RESIDENT BLISS32
ADD/OPEN/SHARED MACRO32
ADD/OPEN DIRECTORY
```

For more information on installing images as known images, see Chapter 8 and the description of the Install Utility in the *VAX/VMS Utilities Reference Volume.*

2.3.6 Running the System Dump Analyzer

Each time the system is bootstrapped, you should run the System Dump Analyzer (SDA) in case the system failed the last time it was running:

```
$ ANALYZE/CRASH_DUMP SYS$SYSTEM:SYSDUMP.DMP
COPY SYS$ERRORLOG:SYSDUMP.DMP
SET OUTPUT LPAO:SYSDUMP.LIS
SHOW CRASH
SHOW STACK/ALL
SHOW SUMMARY
SHOW PROCESS/PCB/PHD/REGISTERS
EXIT
```

If you require further information, you can invoke the System Dump Analyzer for an interactive session upon completion of startup. (See the *VAX/VMS Utilities Reference Volume.*)

Caution: **If you use the page file for the crashdump file, you must, when the system reboots, issue the SDA command COPY to copy the dump from the page file to another file suitable for analysis. If you fail to perform the copy operation, pages used to save the crashdump information will not be released for paging, and your system will hang while executing STARTUP.COM in the rebooting process.**

2.3.7 Maintaining the Operator's Log File

Chapter 10 offers some suggestions for maintaining the operator's log file. If you decide not to put them into practice, you might instead add the following command to purge all but the last two versions of the operator's log file:

```
$ PURGE/KEEP=2 SYS$MANAGER:OPERATOR.LOG
```

2.3.8 Submitting Standard Batch Jobs

Some sites may have batch jobs that are submitted at system startup time and that resubmit themselves to run at intervals as long as the system is running. For such jobs, the DCL command SUBMIT is used in the startup file:

```
$ SUBMIT SYS$MANAGER:file-spec
```

2.3.9 Connecting Devices and Multiport Memory Units

You run the System Generation Utility (SYSGEN) to connect devices not automatically connected in STARTUP.COM and to initialize or connect multiport memory units:

```
$ RUN SYS$SYSTEM:SYSGEN
SHARE MPM1 SHR_MEM_1 /INITIALIZE
```

For installations that require permanent residence of the console block storage device, you must explicitly connect the device by issuing the following SYSGEN command:

```
CONNECT CONSOLE
```

See Chapter 11 and the *VAX/VMS Utilities Reference Volume* for a detailed discussion of SYSGEN.

You should also mount the console block storage device with a command like the following:

```
$ MOUNT/FOREIGN/SYSTEM/PROTECTION=(SYSTEM:RWLP) CSA1:  CONSOLE
```

Failure to mount the console block storage device with appropriate protection permits users to mount and access it, because the device is in RT–11 format and has no file protection.

2.3.10 Installing Secondary Paging and Swapping Files

You run SYSGEN to install secondary paging and swapping files:

```
$ RUN SYS$SYSTEM:SYSGEN
INSTALL DISK$SYS2:[SYSTEM]PAGEFILE2.SYS /PAGEFILE
INSTALL DISK$SYS2:[SYSTEM]SWAPFILE2.SYS /SWAPFILE
```

2.3.11 Providing Announcements

The last command in SYSTARTUP.COM typically announces to all terminals that the system is up and running:

```
$ REPLY/ALL/BELL "VAX/VMS System Initialized"
```

Before SYSTARTUP.COM exits, you may want to provide site-specific definitions for one or both of the following logical names: SYS$ANNOUNCE and SYS$WELCOME. These logical names are checked whenever a user logs in and provide special messages at that time.

2.3.11.1 SYS$ANNOUNCE

SYS$ANNOUNCE defines text to be printed whenever a user begins to log in; that is, the text is printed immediately after a successful dialin, CTRL/Y, or RETURN is received. The text may consist of up to 63 characters. For longer messages, you can precede the name of a text-containing file with an at sign (@) so that the login command procedure prints the entire file as an announcement.

For example, you could include a command of the following form in your SYSTARTUP.COM file:

```
$ DEFINE/SYSTEM SYS$ANNOUNCE "ENTER IF YOU DARE"
```

Or you might prefer to print a file:

```
$ DEFINE/SYSTEM SYS$ANNOUNCE "@SYS$MANAGER:ANNOUNCE.TXT"
```

If you do not define SYS$ANNOUNCE, no announcement is printed.

2.3.11.2 SYS$WELCOME

SYS$WELCOME defines text to be printed whenever a user succeeds in logging in; that is, the text is printed immediately after the correct password is entered. The text may consist of up to 63 characters. For longer messages, you can precede the name of a text-containing file with an at sign (@) so that the login command procedure prints the entire file as a welcoming announcement.

For example, you could include a command of the following form in your SYSTARTUP.COM file:

```
$ DEFINE/SYSTEM SYS$WELCOME "WELCOME TO YOUR OLD VAX/VMS HOME"
```

Or you might prefer to print a file:

```
$ DEFINE/SYSTEM SYS$WELCOME "@SYS$MANAGER:WELCOME.TXT"
```

If you do not specifically define SYS$WELCOME, the standard VAX/VMS welcome message is printed:

```
Welcome to VAX/VMS Version 4.0
```

Note that this message may end by specifying the DECnet–VAX node name if the logical name SYS$NODE is defined.

2.3.12 Redefining the Number of Interactive Users

If the number of interactive users that your site permits to log in at one time is not 64, include the following command in SYSTARTUP.COM:

```
$ STARTUP$INTERACTIVE_LOGINS == n
```

Specify the appropriate number of users for n.

When SYSTARTUP.COM terminates, control is returned to STARTUP.COM, which checks whether the number of logins was set by the above command. If the command was used to specify a value, that value is used. Otherwise, STARTUP.COM sets the number to 64 by default and then exits.

2.3.13 Terminating SYSTARTUP.COM

Include the command EXIT as the last command in SYSTARTUP.COM. The EXIT command terminates the procedure and returns control to STARTUP.COM.

2.4 Selecting a Default Bootstrap Command Procedure

Default bootstrap command procedures are grouped by types of disk controller systems: Integrated Disk Controllers (IDC), UNIBUS Disk Adapters (UDA), UNIBUS controllers, Hierarchical Storage Controllers (HSC), and MASSBUS controllers. In all cases, the bootstrap command procedures are stored on your console media: floppy diskettes for VAX–11/780, or TU58 tape cassettes for VAX–11/750 and VAX–11/730.

After your system is installed, you must select a default bootstrap command procedure from those available on your console volume. To determine which procedures are available, follow these steps:

1. Be sure your console device is connected. If it is not, invoke SYSGEN and issue the following command to connect it:

   ```
   SYSGEN> CONNECT CONSOLE
   ```

2. Next, exit from SYSGEN and mount the console volume as foreign (it is in RT–11 format). In this example, the console volume is assumed to be physically located in the floppy drive (CSA1) on a VAX–11/780.

   ```
   $ MOUNT/FOREIGN CSA1:
   ```

3. Now invoke the Exchange Utility using the following DCL command:

   ```
   $ EXCHANGE
   ```

4. When the EXCHANGE> prompt appears, enter the following command to obtain a listing of the files on the console volume:

   ```
   EXCHANGE> DIRECTORY CSA1:
   ```

   When the listing is printed, examine it to determine which bootstrap command procedures are available for

bootstrapping system devices. Generally, these are the files that start with the letter D and end with either BOO.CMD (nonstop) or GEN. (conversational).

You may also find files that begin with the letters CI. These are used to bootstrap from HSC disks (see Section 4.2.4).

You can relate each procedure to a disk device by noting the first two letters in the filename, which represent the device code. For example, files beginning with the letters DM are used for bootstrapping RK07 disk devices. (See the discussion of SYSGEN in the *VAX/VMS Utilities Reference Volume* for a list of device codes for the various disk drives.)

The third character in the filename is usually a number identical to the unit number of the associated disk device. For example, DM1BOO.CMD is used for bootstrapping the RK07 assigned unit number 1 on the first UNIBUS controller.

If the third character is a letter (A, B, C, or D), you may use the file to bootstrap devices assigned to the related controller (A is the first controller, B the second, and so on). For example, you use DMBBOO.CMD (third character is B) to bootstrap from RK07 disk drives assigned to the second UNIBUS controller. When you use this type of procedure, you must provide the device unit number either by editing the procedure to deposit the selected number in register R3 (the unit number register), or by using the console DEPOSIT command to put the selected number in R3.

For example, if you want to bootstrap from the RK07 drive assigned unit number 3 on the third UNIBUS controller, you would add this line to the bootstrap file named DMCBOO.CMD:

```
DEPOSIT R3 00000003    !DISK DEVICE UNIT NUMBER
```

2.5 Modifying a Default Bootstrap Command Procedure

Once you have selected a default bootstrap command procedure for your site, you may find it necessary to modify the procedure if

- You want to bootstrap your system from a disk drive other than your usual system device.
- You use CIBOO.CMD or a bootstrap command file that does not specify the disk drive's unit number (see Section 4.2.4).

You cannot modify a file on your console volume because of the way the volume is formatted. You must first copy the file to a disk directory, then modify it, and finally copy it back to the console volume. To perform the copy operations, use the DXCOPY command procedure as follows:

1. Invoke DXCOPY:

   ```
   $ @SYS$UPDATE:DXCOPY
   ```

2. When the command procedure asks whether the console storage medium is mounted, type the letter Y:

   ```
   Is the system console storage medium mounted (Y/N)?: Y
   ```

3. When DXCOPY asks if you want to copy a file from the console storage medium (to your default directory), type Y:

   ```
   Copy from console medium (Y/N)?: Y
   ```

4. When DXCOPY prompts you for it, type the name of the file you want to copy to your default directory:

   ```
   Name of file to be copied?: file-spec
   $
   ```

   DXCOPY copies the file and returns you to command level.

   When the DCL prompt appears, you can edit and rename the default bootstrap command procedure file while it resides in your default directory. If you want to make the modified file your default bootstrap command procedure, rename the file to DEFBOO.CMD using the following command:

   ```
   $ RENAME filename DEFBOO.CMD
   ```

5 Invoke DXCOPY to copy the modified procedure back to the console storage medium:

```
$ @SYS$UPDATE:DXCOPY
```

6 When the command procedure asks whether the console storage medium is mounted, type the letter Y:

```
Is the system console storage medium mounted (Y/N)?: Y
```

7 When DXCOPY asks whether you want to copy from the console medium, type N:

```
Copy from console medium (Y/N)?: N
```

The negative response tells DXCOPY you want to copy the file from your default directory to the console storage medium.

8 When the command procedure asks for it, type the name of the file to be copied from your default directory to the console storage medium. In this example, you type DEFBOO.CMD:

```
Name of file to be copied?: DEFBOO.CMD
$
```

DXCOPY responds by copying the modified file to the console volume and returning you to command level.

You may now use the file to bootstrap your system disk from the selected device.

2.6 Performing Modifications for Special Configurations or Workload Requirements

When you first bootstrap your system, the DIGITAL-supplied command procedure SYS$UPDATE:AUTOGEN.COM sets the values of the system parameters, the sizes of the paging, swapping, and dump files, and the contents of the default installed image list.

AUTOGEN provides the appropriate values based on your hardware configuration and on estimates that AUTOGEN makes for typical workloads. However, you may need to modify the system parameters if you have an unusual hardware configuration or special workload requirements. Procedures for

making such modifications are described in the discussion of AUTOGEN in Chapter 11.

2.6.1 Modifying Dual-RL02 Systems to Handle Crash Dumps

The dual-RL02 system does not provide a dump file for handling crash dumps. Instead, the paging file is used for this purpose. However, a dual-RL02 configuration can utilize the paging file for crash dumps *only* if the paging file has at least 1000 more blocks than the number of physical pages of memory on the system, and the swapping file has at least 1000 blocks.

You may need to enlarge the size of these files in a dual-RL02 system because AUTOGEN creates paging and swapping files of minimal size in order to conserve space. To change the size of these files, refer to the AUTOGEN discussion in Chapter 11.

2.6.2 Redefining the Line Printer on a VAX–11/730

Line printer device names on VAX–11/730 processors that use a DMF32 "combo" board are defined as LC. All other line printer names are defined as LP. To avoid problems with programs and procedures that use the LP designation, it is recommended that the line printer device name be redefined by adding the following commands to your site-specific SYSTARTUP.COM command procedure:

```
$ DEFINE /SYSTEM LP LCA0:
$ DEFINE /SYSTEM LP0 LCA0:
$ DEFINE /SYSTEM LPA LCA0:
$ DEFINE /SYSTEM LPA0 LCA0:
```

2.7 Rebooting to Install System Modifications

You must shut your system down and then reboot it to ensure that all the modifications you make are installed in the system. For example, note that any system parameters that were added by running the AUTOGEN command procedure do not take effect until you reboot the system.

You can shut down and reboot the system with AUTOGEN commands, or you can execute SHUTDOWN.COM using the following procedure. (Chapter 4 describes SHUTDOWN.COM in more detail.) If your system controls are set for automatic restart, and your default bootstrap command procedure is set to bootstrap your system disk, you need only perform the first step.

Caution: For VAX-11/750 systems, be sure the BOOT DEVICE switch on the Processor Control Panel is positioned to select the disk drive that contains the system disk.

1 Type the following command at the DCL prompt:

```
$ @SYS$SYSTEM:SHUTDOWN
```

This shutdown command procedure asks you several questions, such as the number of minutes until system shutdown and the reason for the shutdown. The questions pertain to situations where you are doing an orderly shutdown with users logged in to the system. You may either provide specific answers to these questions, or simply press the RETURN key to enter default values.

When you see the following message, the system is ready to shut down:

```
SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

2 Use CTRL/P to halt the system. When the console prompt prints out (> > >), the system is shut down.

3 Reboot the system by typing BOOT on the the console terminal, or by actuating the **BOOT** switch on the Processor Control Panel.

Note: For the VAX-11/750, put the POWER-ON-ACTION switch in the RESTART/BOOT position.

The system responds with a message in the following format:

```
  VAX/VMS Version 4.0 <dd-mmm-yyyy hh:mm:ss.s>

%%%%%%%%%%% OPCOM, <dd-mmm-yyyy hh:mm:ss.s> %%%%%%%%%%%
Logfile has been initialized by operator _OPA0:
Logfile is SYS$SYSROOT:[SYSMGR]OPERATOR.LOG;1

%SET-I-INTSET, login interactive limit = 64, Current interactive value = 0
  SYSTEM       job terminated at <dd-mmm-yyyy hh:mm:ss.s>
```

2.8 Backing Up the System

Once you have installed and customized your system, DIGITAL recommends that you perform the following operations:

- Back up the console volume.
- Back up the system disk.
- Build and copy a system disk.

Step-by-step procedures are presented in the following sections.

2.8.1 Backing Up the Console Volume

You should make a backup copy of your console volume to protect against corruption of your original. Sections 2.8.1.1 and 2.8.1.2 describe, respectively, the procedures for backing up the console volume on VAX system configurations with a single console storage device (VAX–11/780, VAX–11/750) and for systems with two console storage devices (VAX–11/730 and VAX–11/725).

The operating system provides a command procedure called CONSCOPY.COM in SYS$UPDATE that is designed to help you copy your console volume to a blank one. You may use this procedure with any member of the VAX family. Section 2.8.1.1 describes the use of CONSCOPY.COM on VAX–11/780 and VAX–11/750 processors. If you have a dual-drive console subsystem (currently available only on VAX–11/730 and VAX–11/725 systems), refer to Section 2.8.1.2.

2.8.1.1 Console Volume Backups for Single-Drive Console Subsystems

The procedure described here applies directly to backing up the console volume (floppy diskette media) on a VAX–11/780. However, by making appropriate substitutions, you can also use it to back up the console volume on a VAX–11/750. For example, substitute "VAX–11/750" for "VAX–11/780" and "console TU58" for "console floppy" .

This is a two-step procedure. First you *save* the console floppy files on a scratch disk directory; then you *restore* the files in the scratch directory to an initialized floppy. (You can use the blank floppy diskette included in your distribution kit.)

To use CONSCOPY, proceed as follows:

1. Log in under the system manager's account.

2. At the DCL prompt, enter the following command:

   ```
   $ @SYS$UPDATE:CONSCOPY
   ```

   CONSCOPY execution begins with this announcement:

   ```
    S Y S $ U P D A T E : C O N S C O P Y . C O M
        Save or restore a VMS console medium.
   ```

3. Enter your processor type at the following CONSCOPY prompt:

   ```
   Which kit do you want to build [780, 750 or 730, default 780]:
   ```

   When you enter your response, CONSCOPY processes it, and then explains its operations:

   ```
       A SAVE operation involves copying the console medium to
       an RT-11  virtual volume, which is a Files-11 file that
       is an image of the RT-11 console volume.

       A RESTORE operation involves copying the entire contents
       of a virtual volume to a console medium.
   ```

4. When CONSCOPY asks which operation you want, enter SAVE:

   ```
   Do you want to SAVE or RESTORE your console floppy?: SAVE
   ```

5 Next, CONSCOPY prompts for the name of the *virtual disk* (see explanation above) to which files are to be saved. Press the RETURN key to select the default (SYS$DISK:CONSOLE.DSK):

```
Enter file name of virtual disk [default SYS$DISK:CONSOLE:DSK]: RET
```

6 To verify the copy operation, press the RETURN key in response to the following prompt:

```
Do you want log messages as files are copied? [Y/N, default YES] RET
```

7 CONSCOPY processes your response, and then prompts for the name of the console device. Enter CSA1:.

```
Enter console device drive (DDCU:): CSA1:
```

8 At the following prompt, insert your console volume in the console drive and press RETURN:

```
Put your console floppy into drive _CSA1:,
and type <RETURN> when ready: RET
```

Note: You probably already have your console volume in the drive; this request is more appropriate to the *restore* operation, during which you must mount a target console volume in the drive.

When you indicate you are ready to continue, CONSCOPY mounts the console volume, and then invokes the Exchange Utility to begin the *save* operation. Several EXCHANGE messages are displayed, followed by file header information and a list of the files that are being saved.

After completing the *save* operation, CONSCOPY prompts you to put your console floppy back into the console drive and press RETURN when you are ready. (Again, this action is more appropriate to the *restore* operation.) CONSCOPY then mounts the console volume with SYSTEM and NOWRITE protection and returns you to command level.

The second phase of the console backup is the restoration of the console image from the virtual disk files to RT–11 format on an initialized medium.

1 At the DCL prompt, again invoke CONSCOPY:

```
$ @SYS$UPDATE:CONSCOPY
```

2 CONSCOPY announces itself and asks:

```
Which kit do you want to build [780, 750 or 730, default 780]:
```

Enter the same response as for the *save* operation.

3 After CONSCOPY describes its two operations and asks which one you want, enter RESTORE.

4 WHEN CONSCOPY prompts for the name of the *virtual disk*, enter the same response as for the *save* operation.

5 Next, tell CONSCOPY whether you want to log messages as files are copied.

6 When CONSCOPY prompts for the name of the console device, again enter CSA1:.

7 At the following prompt, insert the target floppy in the console drive, and then press RETURN:

```
Put your console floppy into drive _CSA1:,
and type <RETURN> when ready: RET
```

CONSCOPY mounts the target floppy, and invokes the Exchange Utility to execute the *restore* operation. Several EXCHANGE messages are displayed, followed by file header information and a list of the files that are being saved.

When the *restore* operation is completed, CONSCOPY prompts you to put your original console floppy back into the console drive and to indicate when you are ready to continue by pressing RETURN.

After you press RETURN, CONSCOPY mounts the console volume with SYSTEM and NOWRITE protection and returns you to command level.

2.8.1.2 Console Volume Backups for Dual-Drive Console Subsystems

This section provides a procedure for backing up your console volume if you have a dual-drive console subsystem. The procedure is simplified because of the dual-drive capability. First, you insert your original console TU58 in CSA2, and a scratch TU58 in CSA1. Then, after mounting both devices, you use the DCL command BACKUP to back up the original console TU58 onto the scratch TU58.

You may use the blank TU58 included in your distribution kit if it was not used for another purpose.

1. Log in under the system manager's account.
2. Check that the console TU58 is in CSA2.
3. Place a blank cartridge in CSA1. Be sure that the **WRITE PROTECT** tab on the TU58 cartridge is set to permit writing, that is, slid in the direction of the arrow inscribed on it.
4. Invoke SYSGEN from command level:

   ```
   $ RUN SYS$SYSTEM:SYSGEN
   ```

   When the SYSGEN> prompt appears, enter the following command:

   ```
   SYSGEN> CONNECT CONSOLE
   ```

5. Then exit from SYSGEN:

   ```
   SYSGEN> EXIT
   ```

6. When SYSGEN terminates, it returns you to command level. Mount CSA1 with this command:

   ```
   $ MOUNT/FOREIGN CSA1
   ```

7. When the system confirms the mount, mount CSA2:

   ```
   $ MOUNT/FOREIGN/NOWRITE CSA2
   ```

8. When the system confirms the mount, back up the console TU58 to a blank TU58:

   ```
   $ BACKUP/PHYSICAL/VERIFY CSA2: CSA1:
   ```

9 It takes approximately 30 minutes for BACKUP to complete the operation. When the command level prompt appears, dismount the two cartridges with the following commands:

```
$ DISMOUNT CSA1
$ DISMOUNT CSA2
```

You should use the newly created console cartridge as your console TU58 to verify its correctness. Treat the original as the backup copy.

2.8.2 Backing Up the System Disk

To back up your system disk, DIGITAL recommends that you use stand-alone BACKUP, which employs a subset of Backup Utility qualifiers. (Refer to the *VAX/VMS Utilities Reference Volume* for a complete description of the Backup Utility and its qualifiers.) It is especially important that you understand the functions of the /IMAGE and /PHYSICAL qualifiers before using stand-alone BACKUP.

If your system was not distributed on magnetic tape, you must build a stand-alone BACKUP kit either on console media or on disk. You can then bootstrap stand-alone BACKUP from the console block storage device or from the alternate directory root [SYSE] on a Files–11 disk.

Since stand-alone BACKUP performs rather slowly on TU58 cartridges, you will save time by using it on disk.

The following sections explain how to

- Build a stand-alone BACKUP kit in [SYSE] on Files–11 disks
- Create a command procedure to bootstrap stand-alone BACKUP from [SYSE].
- Bootstrap stand-alone BACKUP from [SYSE]
- Build a stand-alone BACKUP kit on console media
- Bootstrap stand-alone BACKUP from console media

2.8.2.1 Building a Stand-Alone BACKUP Kit on Disk

To build a stand-alone BACKUP kit on disk, you must copy the necessary files from the system disk to a target disk on another disk drive. (The target disk may be allocated and mounted.) Log in as system manager and invoke the STABACKIT.COM command procedure in SYS$UPDATE as follows:

```
$ @SYS$UPDATE:STABACKIT
```

The procedure prompts you for the name of the target device. After you enter the name and press RETURN, STABACKIT.COM copies the files to the the directory [SYSE.SYSEXE] and logs the copy operation at your terminal. (If the directory does not already exist, it is automatically created.) When all the files have been copied, the procedure displays the following message:

```
Kit is complete.
```

2.8.2.2 Creating a Command Procedure to Bootstrap Stand-Alone BACKUP from [SYSE] on Files–11 Disks

You can create a command procedure to bootstrap stand-alone BACKUP from [SYSE] by copying an existing bootstrap command procedure from the console TU58 to your default disk directory and editing it. You may choose any unused name of the form xxxBOO.CMD for the command procedure you copy. However, DIGITAL suggests you use a filename identical to the copied file, except that you change the first letter to an **X**.

For example, if you want to bootstrap stand-alone BACKUP from [SYSE] on the drive named DUA0, create a bootstrap command procedure by modifying DU0BOO.CMD, and naming the new file XU0BOO.CMD.

If you want to use a modified form of the default bootstrap command procedure (DEFBOO.CMD), name the created file XTBBOO.CMD. The following procedure presumes you will use a file named XTBBOO.CMD and explains how to modify it.

1 Insert the console TU58 into CSA2.

2 Invoke the DXCOPY command procedure:

```
$ @SYS$UPDATE:DXCOPY
```

When the procedure asks whether the console storage medium is mounted, type the letter Y:

```
Is the system console storage medium mounted (Y/N)?: Y
```

3 When DXCOPY asks if you want to copy a file from the console storage medium (to your default directory), type Y:

```
Copy from console medium (Y/N)?: Y
```

4 When DXCOPY prompts you for it, type the name of the file you want to copy to your default directory. For example, if you are copying DEFBOO.CMD, you would type:

```
Name of file to be copied?: DEFBOO.CMD
$
```

5 When the DCL prompt appears, you can edit and rename the default bootstrap command procedure file while it resides in your default directory. Use the following command to rename the file:

```
$ RENAME DEFBOO.CMD XTBBOO.CMD
```

6 Invoke a text editor to modify the following line in the command procedure:

For VAX-11/780:

```
D R5 0
```

For VAX-11/750, VAX-11/725, VAX-11/730:

```
D/G/L 5 0
```

7 Modify the line as shown:

For VAX-11/780:

```
D R5 E0000000
```

For VAX-11/750, VAX-11/725, VAX-11/730:

```
D/G/L 5 E0000000
```

The hexadecimal digit **E** represents the alternate disk directory root [SYSE], where you have stored stand-alone BACKUP.

8 Invoke DXCOPY to copy the modified procedure back to the console storage medium:

```
$ @SYS$UPDATE:DXCOPY
```

9 When the procedure asks whether the console storage medium is mounted, type Y:

```
Is the system console storage medium mounted (Y/N)?: Y
```

10 When the procedure asks whether you want to copy from the console medium, type N:

```
Copy from console medium (Y/N)?: N
```

The negative response tells DXCOPY you want to copy the file from your default directory to the console storage medium.

11 When the command procedure asks for it, type the name of the file to be copied from your default directory to the console storage medium. In this example, you type XTBBOO.CMD:

```
Name of file to be copied?: XTBBOO.CMD
```

DXCOPY responds by copying the modified file to the console TU58 and returning you to command level.

2.8.2.3 Bootstrapping Stand-Alone BACKUP from Disk

To bootstrap stand-alone BACKUP from disk using the command procedure you have created, follow these steps:

1 Issue the following command to shut down the system:

```
$ @SYS$SYSTEM:SHUTDOWN.COM
```

The shutdown procedure asks you for the number of minutes until system shutdown, the reason for the shutdown, and whether to spin down the disks.

2 Respond to the questions by pressing the RETURN key. As the shutdown continues, the console terminal prints several shutdown messages. When the following message appears, shutdown is completed:

```
SHUTDOWN COMPLETE---USE CONSOLE TO HALT SYSTEM
```

3 Halt the processor by pressing CTRL/P.

4 When the console terminal prints the console mode prompt (>>>), issue the following command:

```
>>> B XTB
```

If you are *not* using a modified version of the default bootstrap command procedure, enter the first three characters in the name of the procedure you created. For example, if the procedure is XUOBOO.CMD, issue the following command:

```
>>> B XUO
```

2.8.2.4 Building a Stand-Alone BACKUP Kit on Console Media

To build a stand-alone BACKUP kit on console media (floppy diskettes or TU58 cassettes), use STABACKIT.COM as described in the following procedure. While the procedure assumes floppy diskettes for the purpose of illustration, it is also applicable to cassettes except for noted differences. The most notable difference is that "TU58 cassette" must be substituted for "floppy diskette" .

If you have a VAX–11/780, the console media are diskettes. For all other VAX systems, the console media are TU58 cassette cartridges.

To build stand-alone BACKUP kit on floppy diskettes, proceed as follows:

1. Label three blank RX1 floppy diskettes SYSTEM_1, SYSTEM_2, and BACKUP respectively.
2. Log in as system manager and invoke STABACKIT.COM.
3. When STABACKIT prompts you for the name of the target device, enter CSA1:.
4. After you have identified the target device, STABACKIT tells you that three blank floppies are needed to build the kit: two for the stand-alone VMS files, and the third for the BACKUP application image.

   STABACKIT also allow you either to build the VMS files as a system kit without building the application kit (in this case BACKUP), or to build both the system and application kits. Respond to both questions by pressing the RETURN key.
5. Next, STABACKIT gives you the option of using the Bad Block Locator Utility (BAD) to check for bad blocks on the target diskette. BAD requires approximately 15 additional minutes for each diskette, but it may save time by finding

bad blocks that would otherwise cause problems in building the kit.

Note: If you are building the kit on TU58 cassettes, allot approximately two hours to run BAD.

6 When STABACKIT prompts, insert the first system floppy (labeled SYSTEM_1) in the console drive and enter YES when you are ready to continue:

```
Please place the first system floppy diskette in drive _CSA1:.
This volume will receive the volume label SYSTEM_1.
Enter "YES" when ready: YES
```

If you elected to run BAD, STABACKIT displays the following message:

```
Analyzing floppy diskette for bad blocks . . .
SYSTEM_1      mounted on _CSA1:
```

The test takes approximately 15 minutes (40 minutes for TU58s). Upon completion, BAD reports the number of blocks on the diskette and the number that are bad. Then STABACKIT mounts the floppy, creates the directory [SYS0.SYSEXE], and copies to it the set of system files while displaying appropriate messages.

7 When the last file is copied, STABACKIT prompts you to insert the next floppy. Replace the floppy diskette labeled SYSTEM_1 with the floppy diskette labeled SYSTEM_2; then enter YES when you are ready to continue:

```
Please place the first system floppy diskette in drive _CSA1:.
This volume will receive the volume label SYSTEM_2.
Enter "YES" when ready: YES
```

STABACKIT runs BAD (assuming you selected this option), mounts the floppy, and copies any remaining system files to [SYS0.SYSEXE].

8 When the last file is copied, STABACKIT prompts you to insert the third floppy. Replace the floppy diskette labeled SYSTEM_2 with the floppy diskette labeled BACKUP; then enter YES when you are ready to continue:

```
Please place the application floppy diskette in drive _CSA1:.
This volume will receive the volume label BACKUP.
Enter "YES" when ready: YES
```

STABACKIT runs BAD, mounts the floppy, and copies the executable BACKUP image (STABACKUP.EXE) to [SYS0.SYSEXE].

9 When the next message appears (indicating that the kit is built), replace the floppy labeled BACKUP with the console floppy; then enter YES when you are ready to continue:

```
The console volume will be mounted /NOWRITE for protection.
Please replace the original console floppy diskette.
Enter "YES" when ready: YES
```

The procedure responds by mounting the console floppy, reporting the starting and ending times, and then returning you to command level.

2.8.2.5 Bootstrapping Stand-Alone BACKUP from Console Media

For purposes of illustration, the procedure described in this section assumes TU58 cartridges as the console media. However, the procedure is also applicable to diskettes. Simply substitute "system diskette" for "TU58 cartridge" .

To bootstrap stand-alone BACKUP from console media, follow these steps:

1 On VAX–11/780 and VAX–11/750 systems ignore this step.

Insert the console TU58 in the CSA2 drive, and insert the stand-alone volume labeled SYSTEM_1 in the CSA1 drive. (On UDA- or RA80-based VAX–11/730 systems and VAX–11/725 systems, CSA1 is on the right-hand side of the processor control panel. On UDA-based VAX–11/730 systems, CSA1 is the left-hand drive.)

2 If the processor is not in console mode, press CTRL/P.

3 When the console terminal prints the console mode prompt (>>>), issue the following command:

```
>>> B CS1
```

4 You will be prompted to place successively in the console drive the three volumes containing the stand-alone BACKUP kit. The first prompt directs you to replace the console TU58 with the stand-alone volume labeled SYSTEM_1 and to enter YES when you are ready to continue. *On VAX–11/730*

and VAX-11/725 systems, you are prompted to replace the SYSTEM_1 volume in the console device with the first stand-alone volume; ignore the message and just enter YES. When you enter YES (or Y), the procedure displays this message:

```
Resuming load operation on volume 'SYSTEM_1', please stand by...
```

5 When the first volume is booted, the procedure prompts you to replace it with the stand-alone volume labeled SYSTEM_2 and to enter YES when you are ready to continue. When you enter YES, the procedure displays the following message:

```
Resuming load operation on volume 'SYSTEM_2', please stand by...
```

6 When the second volume finishes booting, the stand-alone systems announces itself and asks you to enter the date and time. Enter the information in the format displayed by the prompt and press RETURN.

Note: If applicable, the stand-alone system configures and lists your HSC and MSCP-served devices. If you are building your system on an HSC disk, enter the appropriate device name from this list in place of the variable *system device* when you restore the *required* save set. For detailed information on restoring the save set, refer to the installation booklet for your VAX processor.

7 Next, the procedure prompts you to replace the second volume with the application volume labeled BACKUP and to enter YES when you are ready to continue.

When you enter YES, the procedure displays this message:

```
Resuming load operation on volume 'BACKUP', please stand by...
```

When the application volume is booted, stand-alone BACKUP identifies itself and displays the DCL prompt:

```
%BACKUP-I-IDENT, stand-alone BACKUP V4.0; the date is <dd-mmm-yyyy hh:mm>
$
```

Note: Do not remove the BACKUP cartridge from the CSA1 drive until directed to do so.

When your backups are completed, press CTRL/P to halt the processor and terminate stand-alone BACKUP.

2.8.3 Building and Copying a VAX/VMS System Disk

At some point, you may want to transport your operating system files to another disk. For example, assume that your operating system is initially stored on an RK07 disk together with some of your user files. Assume further that you have purchased an RP06 disk drive and that you want to move only the operating system files from the RK07 to the RP06. You can build the operating system on the RP06 without affecting the user files on the RK07 by using the VMSKITBLD command procedure. You should note that the VMSKITBLD **BUILD** function effectively eliminates any software on the target disk.

Another way to use VMSKITBLD.COM is illustrated in this situation. Assume that you are currently using an RK07 to store operating system files and user files, and that you have an RP06 that you are using for user files. You decide to purchase additional optional software products that layer onto the operating system, but you do not have enough space on the RK07 for the additional software. Here, you would like to move the operating system files to the RP06 *without* affecting the user files on the RP06. In this situation, you can use the **COPY** capability of VMSKITBLD because the **COPY** function does not affect files that are currently on the target disk.

2.8.3.1 Building the Operating System on Another Disk

If you want to build your operating system on another disk, and you are not concerned about losing anything on the target disk, use the VMSKITBLD **BUILD** function as illustrated in the following procedure.

1. Bootstrap the operating system from the source disk, typically an RK07.

2. Log in under the system manager's account.

3. Establish SYS$UPDATE as the default directory:

   ```
   $ SET DEFAULT SYS$UPDATE
   ```

4. Place the target disk (assuming you are using a removable disk) in an appropriate drive and put it on line.

5. Enter the following command to invoke VMSKITBLD.COM:

   ```
   $ @VMSKITBLD
   ```

6 In response to VMSKITBLD prompts, supply the requested information about the source and target disk drives.

7 After you supply the required information, VMSKITBLD asks:

```
Operation [BUILD,ADD,COPY,COMMON]?
```

Enter the word BUILD. (If you want your target disk to be built as a cluster-common system disk, enter COMMON.)

Caution: The BUILD option destroys all previous information on the target disk before it builds the operating system.

When the build resumes, you will receive messages that either prompt you for information needed to complete the operation, or inform you of the procedure's status.

A typical message sequence might look like this:

```
Enter mounted source disk name (DDCU:): SYS$SYSDEVICE:
Enter SOURCE disk top level system directory [default = SYS0]: RET
Enter target disk name (DDCU:): DRA0: RET
Enter the target disk's label [default = VAXVMSRL4]: RET
Enter TARGET disk top level system directory [default = SYS0]: RET
  It will be necessary to initialize the target disk(s).
Is the target disk, DRA0:, ready to be initialized? (Y/N): Y
  _DRA0: allocated
%MOUNT-I-MOUNTED, VAXVMSRL4   mounted on _DRA0:
Create directory entries on the target disk.
Copy the system executive files.
Copy the system library files.
Copy the system message files.
Copy the system manager files.
Copy the system update command files.
Copy the system EXE files.
Copy the system help files.
Write a bootblock.
Copy BLISS require files and STRLET DCL library.
Copy coding examples.
Copy the UETP files.
```

VMSKITBLD.COM informs you when the system disk is built by sending the following message to your terminal:

```
Kit is complete.
```

8 At this point, the target disk contains all the required VAX/VMS files for a complete system. You can configure the system properly by bootstrapping with the default parameters and then using the AUTOGEN procedure to configure the target system.

If you have a VAX–11/780, bootstrap the system with the default parameters, using the GEN version of the bootstrap command file:

```
>>> @dduGEN
```

If you have a VAX–11/750, enter:

```
>>> B/1 ddcu
```

9 When the SYSBOOT> prompt appears, enter the following commands:

```
SYSBOOT> USE DEFAULT
SYSBOOT> CONTINUE
```

10 After the system bootstraps, log in as system manager and run the AUTOGEN procedure described in Chapter 11.

2.8.3.2 Copying the Operating System Files to Another Disk

You can also use VMSKITBLD.COM to copy the operating system files to a target disk without affecting the files that are already on it. But first, your VAX/VMS system must be running and the source disk that you intend to copy must be mounted. Often, this source disk is the system disk from which the system was bootstrapped. Proceed as follows to copy the source disk to a target disk:

1 Log in under the system manager's account.

2 Establish SYS$UPDATE as the default directory:

```
$ SET DEFAULT SYS$UPDATE
```

3 Place the target disk on an appropriate drive.

4 Enter the following command to invoke VMSKITBLD:

```
$ @VMSKITBLD
```

5 In response to VMSKITBLD prompts, supply the requested information about the source and target disk drives.

6 After you supply this information, VMSKITBLD asks:

```
Operation [BUILD,ADD,COPY,COMMON]?
```

Enter the word COPY.

7 After you reply, you will receive messages that either prompt you for information needed to complete the copy operation, or inform you of the procedure's status.

8 When the copy operation is finished, VMSKITBLD sends you the following message:

```
Kit is complete.
```

3 Customizing Small-Disk Systems

This chapter discusses the VAX/VMS small-disk environment, including supported configurations, operating system file groups, the VAX/VMS Tailoring Facility (VMSTAILOR), and operating procedures peculiar to small-disk systems. Error and informational messages from the facility are also described.

Currently, DIGITAL supports the following small-disk systems:

- VAX–11/725 systems
- VAX–11/730 systems with dual-RL02 disk drive configurations
- Systems with dual-RK07 disk configurations

During an installation or upgrade procedure, you indicate that you want to generate a tailored system by responding to the appropriate prompts. You can then use the Tailoring Facility to customize your system for specific applications.

Note: Tailoring is presently supported only on system configurations that have less than 140,000 blocks of disk space.

In small-disk systems, a library disk supplements the system disk. At any instant, the system disk need include only those portions of the operating system that are necessary to perform a particular task. Whenever you want to tailor your system disk for a certain task, you use tailoring commands to copy the appropriate operating system files from the library disk to the system disk and to remove from the system disk any files that are not needed. In this way, you obtain additional space for user files on the system disk. (Tailoring commands are described in Section 7.3.)

Note that the size of the library disk must be at least 20,000 blocks (RL02). Moreover, if it is less than 30,000 blocks, there will be insufficient space for the optional save set, which includes the system map, BLISS require files, and the SYS$EXAMPLES directory.

3.1 DIGITAL-Supplied File Groups

To facilitate tailoring procedures in small-disk environments, operating system files are divided into groups. A file group is a file that lists each file in a particular group in the following format:

```
[directory]name.type
```

DIGITAL supplies a standard set of file groups with small-disk systems. For added flexibility, you can create your own file groups, using the format of the DIGITAL-supplied file groups. This section describes the file groups provided by DIGITAL; the next section outlines procedures for creating site-specific file groups.

Files provided by DIGITAL on the library disk are divided into task-related groups. For example, the files used to manage queues are collected in a file group called QUEUES. When you want queue management capability, you specify QUEUES as the parameter for the tailoring COPY command. VMSTAILOR then copies all files in the QUEUES file group from the library disk to the system disk.

Conversely, if you do not need a particular operating system function, you can remove the related file group from the system disk. For example, after you run the User Environment Test Package (UETP), you can remove the operating system files that support this function by specifying UETP as the parameter in a tailoring DELETE command.

Following is a list with a brief description of the DIGITAL-supplied file groups. The approximate size of each group, in blocks, is included in the list to help you make tailoring decisions.

SYSTEM FILE GROUP

- REQUIRED—files needed to boot the VAX/VMS operating system and to perform basic file manipulations, such as COPY, RENAME, and TAILORING, and to provide all the features of the VMS executive. Support for all standard VAX processors and devices is included. These files should not be removed from the system disk. (3300 blocks)

LIBRARY FILE GROUPS

- BLISSREQ—source files used to generate compile-time libraries during VAX BLISS language installation. These files may be deleted after the BLISS language installation is complete. (2800 blocks)
- DECNET—files that give the system a local networking capability. However, you must purchase a separate kit and license to gain multinode networking capability. (1100 blocks)
- DEVELOP—files used for the development of application programs. They include the EDT editor, MACRO–32 assembler, VAX/VMS Librarian, VAX/VMS Linker, VAX/VMS Symbolic Debugger, and the libraries that they use. Help text for these utilities is also included. (8200 blocks)
- EXAMPLES—programming examples for various VAX/VMS applications, along with the system map. These examples are also reproduced in the microfiche source listings. (1500 blocks)
- FILETOOLS—utilities used to analyze and dump files and to create and modify ISAM files. (600 blocks)
- HELP—help text displayed by the VMS HELP command and various system utilities. (4400 blocks)
- LIBRARY—libraries used during the assembly, linking, and compilation of programs. (5000 blocks)
- MANAGER—files used to perform system management functions. They include accounting, disk quota, volume preparation and maintenance, and user authorization functions, as well as command procedures used by the system manager. (1500 blocks)
- MISCTOOLS—system programming support tools, tools for analyzing crash dumps, and other miscellaneous tools. (2200 blocks)
- QUEUES—files that are used to initialize, start, and stop the batch and device queues, and to submit jobs. (200 blocks)
- TEXTTOOLS—files used for text editing and text formatting. They include all editors and DIGITAL Standard Runoff. (580 blocks)

- UETP—User Environment Test Package files, which provide a variety of tests to ensure that your VMS system is operating properly. (494 blocks)

To obtain a list of files in a file group, invoke the Tailoring Facility with the following DCL command:

```
$ @SYS$UPDATE:VMSTAILOR
```

When the system responds with the Tailor> prompt, you can request, for example, a list of files in the QUEUES file group:

```
Tailor> DIRECTORY QUEUES
Contents of QUEUES Group
---------------------------
[SYSEXE]INPSMB.EXE
[SYSEXE]PRTSMB.EXE
[SYSEXE]QUEMAN.EXE
[SYSEXE]SUBMIT.EXE
[SYSEXE]SMBSRVSHR.EXE
```

3.2 Creating Site-Specific File Groups

You can create file groups for your own site-specific applications by following the format of the DIGITAL-supplied file groups. Be sure, however, not to modify the file groups provided by DIGITAL, because they are used for system updates and upgrades as well as for optional software product installations.

You may use any text editor to create a file that lists a file group. If you create a new file group by modifying a copy of a DIGITAL-supplied file group, be sure to assign a new filename to the file you create, and be careful not to modify the DIGITAL-supplied file. For example, if you want to create your own text-processing file group by editing SYS$UPDATE:TEXTTOOLS.TLR with the EDT editor, the following command will ensure that the new file has a new name and that the DIGITAL-supplied file is not altered:

```
$ EDIT /OUTPUT=new_file_name SYS$UPDATE:TEXTTOOLS.TLR
```

Since the DIGITAL-supplied system file group files reside in the SYS$UPDATE directory on the system disk and have the file type TLR, you need not include the device, directory, or file type when you refer to them in tailoring commands.

If you want to put the file groups you create in a different directory or on a different device, you must explicitly specify the files in the tailoring commands. For example, if you create a file group named SPECIAL.TLR and store it in directory [USER.JOBS] on device USER$DISK, use the following command to copy it to the system disk:

```
Tailor> COPY USER$DISK:[USER.JOBS]SPECIAL
```

3.3 Tailoring Commands

The Tailoring Facility provides a set of commands and qualifiers you use to tailor your system disk. The key to tailoring is knowing which file groups you need to support your system operations, and then copying only those groups to the system disk.

There are two ways to issue tailoring commands. You can either invoke the facility and enter commands at the tailoring command level, or you can, with a single DCL command, specify a particular tailoring function.

To work from the tailoring command level, use the following procedure:

1 Log in under the system manager's account.

2 At the DCL prompt, enter the following command:

   ```
   $ @SYS$UPDATE:VMSTAILOR
   ```

   The system responds with the Tailor> prompt to indicate that you are at the tailoring command level.

To return to DCL level, use the tailoring command EXIT.

If you want to tailor from DCL level, enter a single-line DCL command to invoke tailoring and specify the tailoring function you want. For example, if you are at DCL level and you want to copy the TEXTTOOLS file group to the system disk, first make sure that the library disk has been mounted with the tailoring command MOUNT, and then enter the following command:

```
$ @SYS$UPDATE:VMSTAILOR COPY TEXTTOOLS
```

DCL invokes the Tailoring Facility, which executes the command and returns you to DCL level.

Table 3–1 summarizes the VMSTAILOR commands. When you use tailoring commands, observe the following rules:

- The commands that use file group names as parameters accept any unique abbreviation of a name. However, if you have created a file group that resides in a device and directory other than SYS$UPDATE, you will have to be explicit in identifying the device and directory when you specify the file group name as a parameter.
- For the COPY, DELETE, DIRECTORY/SIZE, and RECORD commands to execute, the library disk must be mounted with the tailoring command MOUNT.
- The library disk should be used as a read-only source of VAX/VMS file groups. In order to support system upgrades and updates, its contents should not be altered. Mount the library disk with the /WRITE qualifier only during maintenance updates or layered product installations. (VMSINSTAL, for example, mounts the library disk /WRITE.)
- The /LIBRARY qualifier is reserved for use by the command procedures that are used to do updates and upgrades. You should never use this qualifier when you are tailoring your system.

Caution: If you alter your library disk, you will produce an unsupported software configuration that may result in incorrect system operation.

Table 3–1 VMSTAILOR Command Summary

Command	Function
COPY file-group[,...]	Copies the specified file group(s) from the library disk to the system disk
DELETE file-group[,...]	Deletes previously copied file group(s) from the system disk
DIRECTORY file-group[,...]	Gives a directory of the files in the specified file groups
DISMOUNT [ddcu:]	Dismounts the library disk from the specified device
EXIT	Exits from the Tailoring Facility
HELP [topic] [subtopic]	Explains the specified tailoring commands and qualifiers
MOUNT ddcu: [label]	Mounts the library disk on the specified device
RECORD match [revised]	Creates a file listing all VAX/VMS files found on the library disk that have a match on the system disk
SEARCH file-spec	Locates the file group(s) in which the specified file resides

COPY Command

Copies file groups or individual files in the specified file groups from the library disk to the system disk. The copied files replace any older versions. The COPY command takes the form:

```
COPY file-group[,...]
    /CONFIRM
    /FILE
    /LIBRARY
    /LOG
```

file-group

Specifies one or more file groups or an individual file to be copied.

/CONFIRM

Asks you to confirm that you want each file in the group copied. If you respond with N, the associated file is not copied. If you reply Y or press the RETURN key, the file is copied.

/FILE

Indicates that an individual file is to be copied instead of a file group. You must use a file specification in place of the file group parameter with this qualifier. A default device and directory of LIB$SYSROOT:[SYSEXE] and a default file type of EXE are assumed. When you use the /LIBRARY qualifier with the /FILE qualifier in update and upgrade command procedures, a default device and directory of SYS$SPECIFIC:[SYSEXE] and a default file type of EXE are assumed. You may use wildcards in the file specification, but not lists of file specifications.

/LIBRARY

This qualifier is reserved for use by update and upgrade command procedures to copy files from the system disk to the library disk. Do not specify this qualifier when using tailoring commands interactively.

/LOG

Displays the name of each file that is copied.

Example

The following command copies files in the QUEUES and FILETOOLS groups from the library disk to the system disk and displays the name of each file at the terminal.

```
Tailor> COPY/LOG QUEUES,FILETOOLS
```

The facility responds by listing the files that are copied.

If the system disk does not have enough space to copy a file group, you receive the following message:

```
%BACKUP-E-OPENOUT, error opening SYS$SPECIFIC:[SYSEXE]EDF.EXE;1 as output
-RMS-F-FUL, device full (insufficient space for allocation)
```

If this happens, you can use the DELETE command to remove unnecessary files from the system disk and then try copying again.

DELETE Command

Deletes all versions of each file in the specified file groups from the system disk. The command checks for a backup copy on the library disk before doing the deletion. The DELETE command takes the form:

```
DELETE file-group[,...]
    /CONFIRM
    /FILE
    /LIBRARY
    /LOG
    /OVERRIDE
```

file-group

Specifies one or more file groups or an individual file to be deleted from the system disk. The REQUIRED file group cannot be deleted using this command.

/CONFIRM

Asks for confirmation before deleting each file. If you enter N or press the RETURN key, the file is not deleted. If you enter Y, the file is deleted.

/FILE

Indicates that an individual file is to be deleted instead of a file group. You must use a file specification in place of the file group parameter with this qualifier. A default device and directory of LIB$SYSROOT:[SYSEXE] and a default file type of EXE are assumed. When you use the /LIBRARY qualifier with the /FILE qualifier in update and upgrade command procedures, a default device and directory of SYS$SPECIFIC:[SYSEXE] and a default file type of EXE are assumed. You may use wildcards in the file specification, but not lists of file specifications.

/LIBRARY

This qualifier is reserved for use by update and upgrade command procedures to delete files from the library disk. Do not specify this qualifier when using tailoring commands interactively.

/LOG

Displays the name of each file that is deleted.

/OVERRIDE

Overrides the check for a backup copy before deleting. This qualifier permits you to delete files on the system disk without having to mount the library disk.

Example

The following command deletes the files in the LIBRARY group from the system disk.

```
Tailor> DELETE/CONFIRM LIBRARY
```

The facility then displays the name of each file that you may elect to delete and asks you to confirm the deletion by entering Y:

```
SYS$SPECIFIC:[SYSEXE]MP.STB;1 delete? (Y or N) : Y
SYS$SPECIFIC:[SYSEXE]SYS.STB;1 delete? (Y or N) : Y
SYS$SPECIFIC:[SYSEXE]SYSDEF.STB;1 delete? (Y or N) : Y
SYS$SPECIFIC:[SYSLIB]FORDEF.FOR;1 delete? (Y or N) : Y
SYS$SPECIFIC:[SYSLIB]FORIOSDEF.FOR;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]IMAGELIB.OLB;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]LIB.MLB;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]LIBDEF.FOR;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]MTHDEF.FOR;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]SIGDEF.FOR;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]STARLET.MLB;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]STARLET.OLB;1 delete? (Y or N) : N
SYS$SPECIFIC:[SYSLIB]XFDEF.OLB;1 delete? (Y or N) : N
```

DIRECTORY Command

Lists the contents of the specified file groups. The DIRECTORY command takes the form:

```
DIRECTORY file-group[,...]
    /GROUPS
    /OUTPUT=file-spec
    /SIZE
```

file-group

Specifies the file group.

/GROUPS

Lists all the file groups available in the SYS$UPDATE directory. No parameters are allowed with this qualifier.

/OUTPUT=file-spec

Requests that the output listing be written to the specified file rather than to the current SYS$OUTPUT device. The default file type is LIS.

/SIZE

Requests that the size of each file be displayed along with its name.

Example

The following command lists the files in the HELP and REQUIRED groups in an output file called HELREQ.LIS.

```
Tailor> DIRECTORY/OUTPUT=HELREQ HELP,REQUIRED
```

DISMOUNT Command

Dismounts the library disk from the specified device and deassigns the logical names defined by VMSTAILOR when the disk was mounted. Table 3–2 lists these logical names.

Table 3–2 Library Disk Logical Name Definitions

Logical Name	Definition
LIB$SYSDEVICE	= ddcu:
LIB$TOPSYS	= SYS0
LIB$SYSROOT	= ddcu:[SYS0.]
LIB$ERRORLOG	= LIB$SYSROOT:[SYSERR]
LIB$HELP	= LIB$SYSROOT:[SYSHLP]
LIB$LIBRARY	= LIB$SYSROOT:[SYSLIB]
LIB$MAINTENENCE	= LIB$SYSROOT:[SYSMAINT]
LIB$MANAGER	= LIB$SYSROOT:[SYSMGR]
LIB$MESSAGE	= LIB$SYSROOT:[SYSMSG]
LIB$SHARE	= LIB$SYSROOT:[SYSLIB]
LIB$SYSTEM	= LIB$SYSROOT:[SYSEXE]
LIB$TEST	= LIB$SYSROOT:[SYSTEST]
LIB$UPDATE	= LIB$SYSROOT:[SYSUPD]

The DISMOUNT command takes the form:

```
DISMOUNT [ddcu:]
```

ddcu:

Specifies the drive from which you want to dismount the library disk. If you do not specify a device, the logical name LIB$SYSDEVICE: is used as the default.

EXIT Command

Exits from the Tailoring Facility and returns you to DCL level. The EXIT command takes the form:

```
EXIT
```

HELP Command

Gives detailed descriptions of all the tailoring commands and qualifiers. If you type only HELP, you will receive an explanation of the HELP command and a list of other topics described in the Help file. The HELP command takes the form:

```
HELP [topic] [subtopic]
```

topic

Specifies any of the commands, plus the additional topic of GROUPS.

subtopic

Can be a qualifier when used with a command topic, or a group name when used with the GROUP topic.

Example

The following command displays information about the DEVELOP file group.

```
Tailor> HELP GROUPS DEVELOP
```

The system responds with the following information:

```
GROUPS
  DEVELOP
     Develop files are used for the development of application
     programs.  They include the EDT editor, MACRO-32 assembler,
     VAX/VMS librarian, VAX/VMS linker, VAX-11 Symbolic Debugger, and
     the libraries they use.  Help text for the mentioned utilities
     is also included.
```

MOUNT Command

Mounts the library disk on the specified disk drive, and defines appropriate logical names (see Table 3–2). To use the COPY, DELETE, DIRECTORY/SIZE, and RECORD commands, you must have mounted the library disk with the MOUNT command. The MOUNT command takes the form:

```
MOUNT ddcu: [label]
    /WRITE
```

ddcu:

Specifies the drive on which you wish to mount the library disk.

label

Specifies the label of the library disk you wish to mount. VAXVMSLB4 is the default label.

/WRITE

Lets you write to the library disk. Use the /WRITE qualifier only if you are installing optional software or if you want to run the UETP.

Caution: If you alter your library disk, you will produce an unsupported software configuration, and you may adversely affect system operation. The library disk is write protected by default.

Example

The following command mounts the library disk in the DQA1 drive:

```
Tailor> MOUNT DQA1:
```

RECORD Command

Creates a file group containing names of all the VMS files on the library disk that currently match those on the system disk. With this command, you can save a record of a unique group of files for later use. The RECORD command takes the form:

```
RECORD match-file-group [revised-file-group]
```

match-file-group

Provides a name for the match-file group you want to create. Both the name and revision date of a file must match on both disks for the file to be listed in this file group. If multiple versions of a file exist, only the latest version is compared and listed.

revised-file-group

Provides a name for the revised-file group you want to create. This group includes the files on the system disk that have a matching name on the library disk but have a different revision date. If you do not specify a revised-file group and there are revised files on the system disk, you receive an error message for each revised file.

A RECORD command is typically followed by a DELETE command to allow another software configuration to be created on the system disk. Then, the file group created by the RECORD command can be copied back to the system disk at a later date, restoring the old configuration.

Examples

To list the current tailored software configuration in a file group called MYCONFIG, invoke VMSTAILOR, mount the library disk using the MOUNT command, and issue the following RECORD command:

```
Tailor> RECORD MYCONFIG
```

Now delete MYCONFIG from the system disk, using the DELETE command.

```
Tailor> DELETE MYCONFIG
```

You can then build a new system disk configuration by copying other file groups to the system disk. For example, if you wish to create an environment in which system management functions can be performed, you would issue the following command:

```
Tailor> COPY MANAGER
```

Now you can dismount the library disk with the tailoring DISMOUNT command, and perform system management operations. When you want to restore the previous configuration, remount the library disk and type the following commands:

```
Tailor> DELETE MANAGER
Tailor> COPY MYCONFIG
```

SEARCH Command

Locates the file group(s) in which a specified file resides. The SEARCH command takes the form:

```
SEARCH file-spec
```

file-spec
Specifies the name of a file to be located. No wildcards are allowed. Output is written to SYS$OUTPUT.

3.4 Special Operations for Small-Disk Systems

The space limitations of a small-disk system require you to perform certain operations differently than you would in a large-disk environment. These operations are:

- Running the UETP
- Enabling job queues
- Analyzing system failures
- Installing image files with privileges
- Copying to the system disk image files needed for DCL commands

The following sections describe procedures for performing these operations.

3.4.1 Preparing to Run the UETP

The User Environment Test Package (UETP) tests your hardware and software to verify that your system has been properly installed. The UETP can be run either from the library disk or after being copied to the system disk.

Both methods require you to copy additional files to the system disk. Use the following command to copy the files:

```
$ @SYS$UPDATE:VMSTAILOR COPY DEVELOP,FILETOOLS, -
_$ TEXTTOOLS,MISCTOOLS
```

If you do not have sufficient space to copy all of the file groups, you may instead copy the required individual files. To do so, invoke tailoring with the following command:

```
$ @SYS$UPDATE:VMSTAILOR
```

Now enter the following sequence of tailoring commands:

```
Tailor> COPY /FILE CONVERT
Tailor> COPY /FILE DIFF
Tailor> COPY /FILE SEARCH
Tailor> COPY /FILE SORT32
Tailor> COPY /FILE [SYSLIB]CONVSHR
Tailor> COPY /FILE [SYSLIB]FDLSHR
Tailor> COPY /FILE [SYSHLP]HELPLIB.HLB
Tailor> EXIT
```

When you have completed these preparations, run the UETP according to the instructions in the *Guide to VAX/VMS Software Installation.*

The error message "%JBC-E-SYMBDSAB, symbiont manager is disabled" may appear during execution of the load test phase of the UETP if you have not enabled job queues on your system. This message can be ignored. See Section 3.4.2 for more information about enabling job queues.

3.4.1.1 Running the UETP from the Library Disk

If you plan to run the UETP from your library disk, use the following procedure to alter the default device in the SYSTEST user authorization record to LIB$SYSROOT. This procedure lets the UETP write to your library disk.

If you decide to run the UETP from the system disk at some future time, you will have to reset the device field for the SYSTEST account to SYS$SYSROOT before doing so.

1. Log in under the system manager's account.
2. Issue the following command:

   ```
   $ RUN SYS$SYSTEM:AUTHORIZE
   ```

3. Then enter responses to the UAF> prompt as follows:

   ```
   UAF> MODIFY SYSTEST/DEVICE=LIB$SYSROOT:
   UAF> EXIT
   ```

4. Place the library disk in the appropriate drive (ddcu:).

5 Invoke tailoring and mount the library disk as writeable with the following DCL command:

```
$ @SYS$UPDATE:VMSTAILOR MOUNT/WRITE ddcu:
```

3.4.1.2 Running the UETP from the System Disk

If you plan to run the UETP from your system disk, you must first mount the library disk with the tailoring command MOUNT, and then copy the UETP file group to your system disk.

1 Place the library disk in the appropriate drive (ddcu:).

2 Invoke tailoring with the following command:

```
$ @SYS$UPDATE:VMSTAILOR
```

3 Enter the following sequence of tailoring commands:

```
Tailor> MOUNT ddcu:
Tailor> COPY UETP
Tailor> EXIT
```

3.4.2 Enabling Job Queues

Job queues are disabled by default. If you wish to create queues, use the following commands to copy the QUEUES tailoring group to your system disk:

```
Tailor> MOUNT ddcu:
Tailor> COPY QUEUE
Tailor> EXIT
```

Then shut down and reboot as described in Section 2.7.

Approximately 100 blocks are required to create the system queue file (SYS$SYSTEM:JBCSYSQUE.EXE). See Chapter 9 for more information about creating and managing queues.

3.4.3 Analyzing System Failures

The VAX/VMS operating system includes a capability for writing crash dumps to your system disk when a serious hardware or software problem occurs. Usually a dump file (SYS$SYSTEM:SYSDUMP.DMP) is required equal in size to the number of pages of physical memory on your system plus four blocks. However, in small-disk systems, crash dumps are written instead to the system paging file (SYS$SYSTEM:PAGEFILE.SYS). Normally the dumps are overwritten each time you boot your system. To preserve crash dumps for analysis, you set the SYSGEN parameter SAVEDUMP to 1 and copy the dump to another disk. These operations are described in the next sections.

3.4.3.1 Setting the SAVEDUMP Parameter

For small-disk systems, the SAVEDUMP parameter is off by default. If you wish to preserve crash dumps on a routine basis, you set the parameter to 1. Note, however, that if your system fails and the parameter is set to 0, you may still preserve the dump by stopping in SYSBOOT when you reboot your system and setting the SAVEDUMP parameter to 1. (See Section 4.2.3 for instructions on booting and stopping in SYSBOOT.)

To set the SAVEDUMP parameter while running interactively, invoke SYSGEN as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
```

Modify the SAVEDUMP parameter by typing the following commands at the successive SYSGEN prompts:

```
SYSGEN> USE CURRENT
SYSGEN> SET SAVEDUMP 1
SYSGEN> WRITE CURRENT
SYSGEN> EXIT
```

Shut down and reboot as described in Section 2.7.

3.4.3.2 Saving the Dump for Analysis

Since a large portion of your paging file becomes unavailable when a dump is saved, it is necessary to free this space as quickly as possible. Your system may perform very slowly, or even fail again, when only a small amount of paging file space is available. If less than 1000 blocks of paging file remain, the SAVEDUMP parameter is ignored, and the dump is deleted immediately.

To ensure that dumps are automatically copied to another disk and that space is reclaimed in the paging file, place the following commands in your site-specific startup command procedure (SYS$MANAGER:SYSTARTUP.COM). (These commands assume the presence of a data disk on device DQA1 with a volume label USERDISK and a directory of [CRASH].)

```
$ MOUNT DQA1: USERDISK
$ ANALYZE /CRASH SYS$SYSTEM:PAGEFILE.SYS
$ COPY DQA1:[CRASH]SYSDUMP.DMP;1
$ EXIT
```

After the disk is mounted, the System Dump Analyzer (SDA) is invoked. SDA then acts as follows to manage a crash dump:

1. Exits immediately when invoked from the startup command procedure if one of the following is true:

   a. The dump file contains no valid dump.

   b. The dump has been previously analyzed.

   c. The dump is the result of an operator-requested shutdown.

2. Frees the paging file space for immediate use at the sucessful completion of a COPY operation.

3. When the dump is copied to an already existing file, replaces that file rather than creating a new one. By explicitly specifying a version number of 1, each new crash dump replaces any older crash dump already saved on the user disk.

If you cannot boot or successfully copy a crash dump, you may boot another system disk, mount the system disk containing the dump as a data disk, and then analyze the crash dump.

3.4.4 Installing Image Files with Privileges

The library files listed in Table 3–3 must be installed with privileges on small-disk systems for proper operation.

Table 3–3 Files Requiring Installation with Privileges

File	File Group
AUTHORIZE.EXE	MANAGER
ANALIMDMP.EXE	DEVELOP
INIT.EXE	MANAGER
SHWCLSTR.EXE	MISCTOOLS
MAIL.EXE	MISCTOOLS
PHONE.EXE	MISCTOOLS
RTPAD.EXE	DECNET
CDU.EXE	MISCTOOLS
MONITOR.EXE	MISCTOOLS
SUBMIT.EXE	QUEUES
DBGSSISHR.EXE	DEVELOP

If you bootstrap your system with these files already tailored to your system disk, they will be installed automatically. If you tailor them to your system disk after bootstrapping, you can install them in one of two ways:

- You may shut your system down, using SYS$SYSTEM:SHUTDOWN, and reboot as described in Section 2.7.
- You may invoke the DIGITAL image installation procedure with the following command:

```
$ @SYS$MANAGER:VMSIMAGES INSTALL VMSIMAGES
```

For further information about installed images, see Chapter 8.

3.4.5 Copying to the System Disk Images Needed for DCL Commands

Many DCL commands activate a related VAX/VMS image. For example, the DCL command LIBRARY normally calls the image LIBRARIAN. In a small-disk environment, you cannot invoke an image that is not presently tailored to your system environment. If you attempt to do so, you will receive one of the following error messages:

```
%DCL-W-ACTIMAGE, error activating image 'name'
-CLI-E-IMAGEFNF, image file not found 'image name'
```

Before invoking the image, you must use the tailoring COPY command to copy the appropriate image file or file group from the library disk to the system disk.

Table 3–4 lists commands that invoke an image, the image activated, and the file group in which it is present. The table also shows how command qualifiers may change the image call.

Note: **The relationship between DCL commands and invoked images shown in the table is specific to Version 4.0 of the VAX/VMS operating system and is provided only as a guide for tailoring your system disk. You should avoid writing programs or procedures that depend on this relationship, because it changes from release to release.**

Table 3–4 DCL Commands, Activated Images, and File Groups

Command	Image	File Group
ACCOUNTING	ACC	MANAGER
ANALYZE	ANALYZOBJ	FILETOOLS
/CRASH_DUMP	SDA[1]	MISCTOOLS
/DISK_STRUCTURE	VERIFY	MANAGER
/ERROR_LOG	ERF	MISCTOOLS
/MEDIA	ANALYZBAD	MANAGER
/PROCESS_DUMP	ANALIMDMP	DEVELOP
/RMS_FILE	ANALYZRMS	FILETOOLS
/SYSTEM	SDA[1]	MISCTOOLS
APPEND	COPY	REQUIRED
ASSIGN	DCL	REQUIRED
/MERGE	QUEMAN	QUEUES
/QUEUE	QUEMAN	QUEUES
BACKUP	BACKUP	REQUIRED
CHECKSUM	CHECKSUM	MISCTOOLS
CONVERT	CONVERT,CONVSHR	FILETOOLS
/RECLAIM	RECLAIM,CONVSHR	FILETOOLS
COPY	COPY	REQUIRED
CREATE	CREATE	REQUIRED
/FDL	CREATEFDL,CONVSHR	FILETOOLS
DEASSIGN	DCL	REQUIRED
/QUEUE	QUEMAN	QUEUES
DEFINE	DCL	REQUIRED
/CHARACTERISTIC	QUEUEMAN	QUEUES
/FORM	QUEUEMAN	QUEUES
DELETE	DELETE	REQUIRED
/CHARACTERISTIC	QUEUEMAN	QUEUES
/ENTRY	QUEMAN	QUEUES
/FORM	QUEUEMAN	QUEUES
/QUEUE	QUEMAN	QUEUES

[1]SYS.STB required (LIBRARY file group).

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
DIFFERENCES	DIFF	TEXTTOOLS
DIRECTORY	DIRECTORY	REQUIRED
DISMOUNT	DISMOUNT,DISMNTSHR	REQUIRED
DUMP	DUMP	FILETOOLS
EDIT	EDT	TEXTTOOLS
/ACL	ACLEDT	FILETOOLS,MANAGER
/FDL	EDF,FDLSHR	FILETOOLS
/SUM	SUMSLP,SUMSHR	TEXTTOOLS
/TECO	TECO	TEXTTOOLS
EXCHANGE	EXCHANGE	MANAGER
HELP	HELP[2]	REQUIRED
INITIALIZE	INIT	MANAGER
/BASE_PRIORITY	QUEMAN	QUEUES
/BATCH	QUEMAN	QUEUES
/BLOCK_LIMIT	QUEMAN	QUEUES
/CHARACTERISTICS	QUEMAN	QUEUES
/CPUDEFAULT	QUEMAN	QUEUES
/CPUMAXIMUM	QUEMAN	QUEUES
/DEFAULT	QUEMAN	QUEUES
/DISABLE_SWAPPING	QUEMAN	QUEUES
/ENABLE_GENERIC	QUEMAN	QUEUES
/FORM	QUEMAN	QUEUES
/GENERIC	QUEMAN	QUEUES
/JOB_LIMIT	QUEMAN	QUEUES
/LIBRARY	QUEMAN	QUEUES
/ON	QUEMAN	QUEUES
/PROCESSOR	QUEMAN	QUEUES
/QUEUE	QUEMAN	QUEUES
/RETAIN	QUEMAN	QUEUES

[2]HELPLIB.LIB required (HELP file group).

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
/SEPARATE	QUEMAN	QUEUES
/START	QUEMAN	QUEUES
/TERMINAL	QUEMAN	QUEUES
/WINDOWS	QUEMAN	QUEUES
/WSDEFAULT	QUEMAN	QUEUES
/WSEXTENT	QUEMAN	QUEUES
/WSQUOTA	QUEMAN	QUEUES
LIBRARY	LIBRARIAN	DEVELOP
LINK	LINK[3]	CRFSHR
MACRO	MACRO32[4],SUMSHR	DEVELOP
MAIL	MAIL,CONVSHR,EDTSHR	MISCTOOLS
MERGE	MERGE	MISCTOOLS
MESSAGE	MESSAGE	MISCTOOLS
MONITOR	MONITOR	MISCTOOLS
MOUNT	VMOUNT,MOUNTSHR	REQUIRED
PATCH	PATCH	MISCTOOLS
PHONE	PHONE	MISCTOOLS
PRINT	SUBMIT	QUEUES
PURGE	DELETE	REQUIRED
RENAME	RENAME	REQUIRED
REPLY	REPLY	REQUIRED
REQUEST	REQUEST	REQUIRED
RUN	DCL	REQUIRED
/ACCOUNTING	RUNDET	REQUIRED
/AST_LIMIT	RUNDET	REQUIRED
/AUTHORIZE	RUNDET	REQUIRED
/BUFFER_LIMIT	RUNDET	REQUIRED
/DELAY	RUNDET	REQUIRED

[3]STARLET.OLB and IMAGELIB.OLB required (LIBRARY file group).
[4]SYS.STB required (LIBRARY file group).

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
/DETACHED	RUNDET	REQUIRED
/DUMP	RUNDET	REQUIRED
/ENQUEUE_LIMIT	RUNDET	REQUIRED
/ERROR	RUNDET	REQUIRED
/EXTENT	RUNDET	REQUIRED
/FILE_LIMIT	RUNDET	REQUIRED
/INPUT	RUNDET	REQUIRED
/INTERVAL	RUNDET	REQUIRED
/IO_BUFFERED	RUNDET	REQUIRED
/IO_DIRECT	RUNDET	REQUIRED
/JOB_TABLE_QUOTA	RUNDET	REQUIRED
/MAILBOX	RUNDET	REQUIRED
/MAXIMUM_WORKING_SET	RUNDET	REQUIRED
/OUTPUT	RUNDET	REQUIRED
/PAGE_FILE	RUNDET	REQUIRED
/PRIORITY	RUNDET	REQUIRED
/PRIVILEGES	RUNDET	REQUIRED
/PROCESS	RUNDET	REQUIRED
/QUEUE_LIMIT	RUNDET	REQUIRED
/RESOURCE_WAIT	RUNDET	REQUIRED
/SCHEDULE	RUNDET	REQUIRED
/SERVICE_FAILURE	RUNDET	REQUIRED
/SUBPROCESS_LIMIT	RUNDET	REQUIRED
/SWAPPING	RUNDET	REQUIRED
/UIC	RUNDET	REQUIRED
RUNOFF	RUNOFF	TEXTTOOLS
/CONTENTS	DSRTOC	TEXTTOOLS
/INDEX	DSRINDEX	TEXTTOOLS
SDL/NOPARSE	SDLNPARSE	DEVELOP
SEARCH	SEARCH	TEXTTOOLS

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
SET	SET	REQUIRED
ACCOUNTING	SET	REQUIRED
ACL	SETSHOACL	FILETOOLS,MANAGER
/EDIT	ACLEDT	FILETOOLS,MANAGER
AUDIT	SET	REQUIRED
BROADCAST	SET	REQUIRED
CARD_READER	SET	REQUIRED
CLUSTER	SET	REQUIRED
COMMAND	CDU	MISCTOOLS
DAY	SET	REQUIRED
DEVICE	SET	REQUIRED
/ACL	SETSHOACL	FILETOOLS,MANAGER
/EDIT	ACLEDT	FILETOOLS,MANAGER
DIRECTORY	SET	REQUIRED
/ACL	SETSHOACL	FILETOOLS,MANAGER
/EDIT	ACLEDT	FILETOOLS,MANAGER
FILE	SET	REQUIRED
/ACL	SETSHOACL	FILETOOLS,MANAGER
/EDIT	ACLEDT	FILETOOLS,MANAGER
HOST	RTPAD	DECNET
LOGINS	SET	REQUIRED
MAGTAPE	SET	REQUIRED
MESSAGE	SETPO	REQUIRED
PASSWORD	SETPO	REQUIRED
PRINTER	SET	REQUIRED
PROCESS	SET	REQUIRED
PROTECTION	SET	REQUIRED
QUEUE	QUEMAN	QUEUES
RESTART_VALUE	QUEMAN	QUEUES
RMS_DEFAULT	SET	REQUIRED

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
TERMINAL	SET	REQUIRED
VOLUME	SET	REQUIRED
WORKING_SET	SET	REQUIRED
SHOW	SHOW	REQUIRED
ACCOUNTING	SHOW	REQUIRED
ACL	SETSHOACL	FILETOOLS,MANAGER
AUDIT	SHOW	REQUIRED
BROADCAST	SHOW	REQUIRED
CLUSTER	SHWCLSTR	MISCTOOLS
CPU	MP	REQUIRED
DEVICES	SHOW	REQUIRED
ERROR	SHOW	REQUIRED
LOGICAL	SHOW	REQUIRED
MAGTAPE	SHOW	REQUIRED
MEMORY	SHOW	REQUIRED
NETWORK	SHOW	REQUIRED
PRINTER	SHOW	REQUIRED
PROCESS	SHOW	REQUIRED
QUEUE	QUEMAN	QUEUES
RMS_DEFAULT	SHOW	REQUIRED
SYSTEM	SHOW	REQUIRED
TERMINAL	SHOW	REQUIRED
USERS	SHOW	REQUIRED
WORKING_SET	SHOW	REQUIRED
SORT	SORTMERGE	MISCTOOLS
START	QUEMAN	QUEUES
/CPU	MP	REQUIRED
STOP	DCL	REQUIRED
/ABORT	QUEMAN	QUEUES
/CPU	MP	REQUIRED

Table 3–4 (Cont.) DCL Commands, Activated Images, and File Groups

Command	Image	File Group
/ENTRY	QUEMAN	QUEUES
/HOLD	QUEMAN	QUEUES
/MANAGER	QUEMAN	QUEUES
/NEXT	QUEMAN	QUEUES
/PRIORITY	QUEMAN	QUEUES
/QUEUE	QUEMAN	QUEUES
/REQUEUE	QUEMAN	QUEUES
/RESET	QUEMAN	QUEUES
SUBMIT	SUBMIT	QUEUES
SYCHRONIZE	QUEMAN	QUEUES
TYPE	TYPE	REQUIRED
UNLOCK	UNLOCK	FILETOOLS

3.5 Error and Informational Messages from the Tailoring Facility

This section lists the error and informational messages issued by the Tailoring Facility. Each message consists of an abbreviation followed by a text message.

AMBIGRP, 'group name' is an ambiguous group name

Explanation: The group-name as entered has too few characters to make it unique.

User Action: Reenter the command with a unique group name.

AMBIGWRD, 'word' is an ambiguous word

Explanation: The abbreviated command or qualifier is ambiguous.

User Action: Reissue the command specifying a long enough spelling to resolve the ambiguity.

BADCOMMA, invalid comma in command

Explanation: A comma has been detected preceding the first parameter in a command.

User Action: Correct the command and reissue.

CTRLY, function aborted by CTRL/Y

Explanation: The function in progress has been interrupted by pressing the CTRL and Y keys simultaneously.

User Action: None.

DEVICEREQ, device must be specified

Explanation: A device name was not specified with this command.

User Action: Reenter the command specifying a device name.

DISMOUNT, device dismounted

Explanation: The library disk has been dismounted.

User Action: None.

EXPNOTFND, expected file 'filename' not found

Explanation: A file with the specified name could not be found during an open operation.

User Action: Examine the referenced directory to check for the existence of the named file. Check the spelling of the file specification.

FILENAMEREQ, a filename must be specified

Explanation: A filename must be specified with this command.

User Action: Reissue the command, specifying a filename.

GROUPREQ, a group must be specified

Explanation: A group name must be specified with this command.

User Action: Reissue the command, specifying a group name.

HELPGROUPS, type DIR/ GROUPS for a list of tailoring groups

Explanation: This is an informational message issued when an invalid group name is used.

User Action: Use the DIRECTORY /GROUPS tailoring command to obtain a list of valid group names.

HELPVERB, type HELP 'command'

Explanation: This is an information message issued when a command is improperly used.

User Action: Use the HELP command to obtain more information.

HLPMOUNTWRT, type HELP MOUNT/WRITE

Explanation: Issued following a write-lock error.

User Action: Use the indicated HELP command to obtain more information about mounting disks writeable.

LIBNOTMOUNT, the LIBRARY disk must be mounted

Explanation: A command has been issued that requires the library disk.

User Action: Mount the library disk and reissue the command.

LIBPROTECT, the LIBRARY disk must be writeable

Explanation: A write-lock error message has occurred on the library disk as the result of a command directing the Tailoring Facility to write to the library disk.

User Action: Check that write protect is disabled, mount the library disk writeable, and reissue the command.

NOBACKUP, no backup copy on device 'device'

Explanation: A request to delete a file has been ignored because the file has no backup copy on the library disk (or on the system disk if the /LIBRARY qualifier has been specified).

User Action: Check that the file exists on the backup disk, and that the filename is properly spelled.

NOHELP, help not available

Explanation: Either the Help Utility or the Tailoring help library could not be located on the system disk or, if mounted, on the library disk.

User Action: Mount the library disk and reissue the command. If the library disk is already mounted, then this error indicates that files have been improperly deleted from one or both disks.

NONZEROREF, open files on device, not dismounted

Explanation: The library disk cannot be dismounted because of open channels to the device.

User Action: Investigate the list of files displayed, using the process ID field, to determine what users or batch jobs have files open. Close the files and reissue the DISMOUNT command.

NOQUAL, no qualifiers allowed on the 'command' command

Explanation: No qualifiers are allowed on the indicated command.

User Action: Check the typing of the command. Type HELP to obtain more information about the syntax of the command.

NOSEARCH, SEARCH not available

Explanation: The SEARCH.EXE file could not be located on the system disk or, if mounted, on the library disk.

User Action: Mount the library disk and reissue the command. If the library disk is already mounted, this error indicates that files have been improperly deleted from one or both disks.

NOSUCHDEV, no such device as 'device'

Explanation: The specified device does not exist.

User Action: Check the spelling of the device name. Use the DCL command SHOW DEVICES to obtain a list of legal devices.

NOSUCHDIR, expected directory 'directory' not found

Explanation: The specified directory was not found.

User Action: Check the spelling of the file specification used in the command. If correct, use the DCL command DIRECTORY to check if the directory exists.

NOSUCHFILE, no files matching the specification 'filename' were found

Explanation: No files could be located that matched the indicated file specification.

User Action: Check the spelling of the filename. Use the DCL DIRECTORY command to look for the indicated files.

NOSUCHGROUP, group file 'group name' not found

Explanation: The specified group could not be found.

User Action: Check the spelling of the group name. Use the DIRECTORY /GROUP tailoring command to obtain a list of groups.

NOTDELETED, 'filename' not deleted

Explanation: The indicated filename was not deleted. This is an informational message that accompanies an error message explaining the reason for the delete failure.

User Action: Investigate the associated error message.

NOWILDCARDS, wildcards not allowed in group names

Explanation: Wildcards (* and %) are not allowed in the specification of group names.

User Action: Reissue the command with an explicit group name.

NOWRITE, device is not write enabled

Explanation: An attempt to mount the library disk /WRITE has failed.

User Action: Check that the disk is not write protected. Retry the command.

OPENIN, error opening 'filename' as input

Explanation: The indicated input file could not be opened.

User Action: Check that the file exists and that you have read access to it.

OPENOUT, error opening 'filename' as output

Explanation: The indicated output file could not be opened.

User Action: Check that the directory exists and that you have write access to it.

REISSUE, correct the situation and reissue the command

Explanation: This informational message is issued in conjunction with other error messages.

User Action: Address the problem described by the associated error message and reissue the command.

REQNOTALLOWED, the REQUIRED file group cannot be deleted

Explanation: An attempt was made to delete the REQUIRED file group.

User Action: None. The REQUIRED file group is needed to boot the system and cannot be deleted.

TOOMANYPARM, too many parameters

Explanation: More than the maximum number of parameters have been given on a command line.

User Action: Check the spelling of the command. If correct, then break the command into several simpler commands having fewer parameters.

UNRECCMC, 'command' is an unrecognized command

Explanation: The indicated command is not a valid command.

User Action: Use the HELP command to obtain a list of valid commands.

UNRECQUAL, 'qualifier' not valid for this command

Explanation: The specified qualifier is not valid for the command issued.

User Action: Check the spelling of the qualifier and the command. Use the HELP command to obtain the proper spellings. Note that the Tailoring Facility checks the entire spelling of keywords, unlike DCL, which checks only the first four characters.

USEEXIT, type EXIT to exit

Explanation: An attempt to terminate tailoring is invalid.

User Action: Use the EXIT command to exit.

USEHELP, type HELP for more information

Explanation: This informational message is issued in association with various error messages.

User Action: Use the HELP command to obtain more information about the command you are using. For example, if an error occurs on a COPY command, then type HELP COPY.

4 Shutdown and Startup

During the course of system operation, you will need to shut down and restart your system to perform various maintenance functions. For example, you may need to modify system parameters or make certain hardware modifications. Also, if the system fails and does not automatically restart, you will need to bootstrap the system manually to restart it. DIGITAL provides several procedures for shutting down and restarting the system. This chapter describes these procedures.

4.1 Shutting Down the Operating System

You can perform three types of shutdown operations:

1. **An orderly shutdown with SYS$SYSTEM:SHUTDOWN.COM**. This procedure shuts down the system while performing housekeeping functions such as disablng future logins, stopping the batch and printer queues, dismounting mounted volumes, and stopping user processes. SHUTDOWN.COM is described in Section 4.1.1.

   SHUTDOWN.COM optionally invokes a site-specific command procedure named SYS$MANAGER:SYSHUTDWN.COM, which you tailor to the needs of your installation. The SYSHUTDWN.COM file is present in the VAX/VMS distribution kit but contains no commands.

2. **An emergency shutdown with OPCCRASH**. If you cannot shut down the system with SHUTDOWN.COM, you run the OPCCRASH emergency shutdown program. OPCCRASH is described in Section 4.1.2.

3. **An emergency shutdown with CRASH**. You use this procedure for an emergency shutdown of the system if OPCCRASH fails. On all VAX processors except the VAX-11/750, the procedure is supplied on the system console medium as a console command program named CRASH.

(See Section 4.1.3.) Section 4.1.4 describes the individual instructions you must enter to obtain equivalent results on a VAX–11/750.

4.1.1 Orderly Shutdown with SHUTDOWN.COM

This section describes the use of SHUTDOWN.COM to shut down the system in an orderly fashion. Do not modify SHUTDOWN.COM; instead, you can add commands to the SYS$MANAGER:SYSHUTDWN.COM command procedure to perform site-specific housekeeping functions.

To execute SHUTDOWN.COM, you must have either the SETPRV privilege or all the following privileges: CMKRNL, EXQUOTA, OPER, SYSNAM, SYSPRV, TMPMBX, and WORLD. If you have the SETPRV privilege, the procedure will automatically assign you the required privileges.

To perform an orderly shutdown of the system, invoke SHUTDOWN.COM from any terminal and any account with the following DCL command:

```
$ @SYS$SYSTEM:SHUTDOWN
```

The procedure then prompts with a series of questions and messages. These are shown below, along with directions for appropriate responses.

```
How many minutes until final shutdown [0]?
```

Enter an integer. If the system logical name SHUTDOWN$MINIMUM_MINUTES is defined, its integer value is the minimum value that you can enter. So, if the logical name is defined as 10, you must specify at least 10 minutes to final shutdown. Otherwise an error message is displayed. The logical name value is used as the default if you do not enter a value. If the logical name is not defined, and you do not enter a value, 0 minutes is used.

```
Reason for shutdown:
```

Enter a one-sentence reason for shutting down the system.

```
Do you want to spin down the disk volumes [No]?
```

Enter YES or NO (Y or N). Note, however, that the system disk cannot be spun down.

```
Do you want to invoke the site-specific shutdown procedure [Yes]?
```

Enter YES or NO.

```
Should an automatic system boot be performed [No]?
```

By default, the system does not automatically reboot. However, if you respond with YES, when the shutdown is complete, an attempt is made to reboot the system automatically. Note that the system can be automatically rebooted only if the appropriate hardware switch is also set on the processor, and if the default bootstrap command file is properly set. See Section 2.4 for a description of how to set the default bootstrap command file.

```
When will the system be rebooted [later]?
```

Enter a time in the format you want printed in the message that will be broadcast to the users. For example, you could specify IMMEDIATELY, or IN 10 MINUTES, or a time such as 2 P.M. or 14:00. If you do not know when the system will be available again, press RETURN to specify "later" as the time when the system will reboot.

```
Shutdown options (enter as a comma-separated list):
 REMOVE_NODE          Remaining nodes in the cluster should adjust quorum
 CLUSTER_SHUTDOWN     Entire cluster is shutting down
 REBOOT_CHECK         Check existence of basic system files
Shutdown options [NONE]
```

The procedure prompts you to specify one or more shutdown options. If your system is a VAXcluster member, all three options are listed. (For more information on cluster shutdown options, see the *Guide to VAXclusters*.) Otherwise, only the REBOOT_CHECK option is listed. Enter REBOOT_CHECK if you wish to verify that a subset of the files necessary to reboot the system after shutdown is present. (If you are certain that the files exist, press RETURN.)

If you select the REBOOT_CHECK option, the procedure will check for the necessary files and notify you if any are missing. (Note, however, that this check is not exhaustive.) You should replace such files before proceeding. If all files are present, this success message appears:

```
%SHUTDOWN-I-CHECKOK, Basic reboot consistency check completed.
```

The following events occur as the shutdown procedure continues, and the corresponding messages are printed on the terminal:

1. A message that requests users to log out is broadcast to all users on the system. This message is broadcast at decreasing time intervals.

2. The system logical name SHUTDOWN$TIME is defined as the absolute time of shutdown. For example, if the value 10 is specified as 12:00 in response to the first question, the logical name is assigned the absolute time value 12:10. By requesting the logical name definition for SHUTDOWN$TIME (with the SHOW LOGICAL command), you can see if a shutdown is in progress or determine the actual time of shutdown. This feature is useful if a user for some reason missed the shutdown broadcast message.

3. At six minutes or less until system shutdown, the terminal from which shutdown was invoked is made an operator's console, all future nonoperator logins are disabled, and if the DECnet network is running, it is shut down.

4. When there is one minute left until system shutdown, batch and device queues and the system job queue manager are stopped.

5. At zero minutes, the site-specific command procedure SYS$MANAGER:SYSHUTDWN.COM is invoked.

6. All user processes are stopped; however, system processes continue. ACPs may delete themselves when their mounted volumes are finally dismounted.

7. For dual-processor systems, the secondary processor is stopped.

8. All installed images are removed.

9. All mounted volumes are dismounted and, if you request it, the disks are spun down. Note, however, that the system disk cannot be spun down.

10. The operator's log file is closed.

11. The program SYS$SYSTEM:OPCCRASH is invoked to shut down the system.

12 If you did not request an automatic reboot, the following message appears on the system console:

```
SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

If you requested an automatic reboot, the system reboots, provided the necessary controls are set.

13 If you are not automatically rebooting, you must halt the system after the above message is printed at the console terminal.

Example 4–1 demonstrates an orderly system shutdown on node AVALON.

Example 4–1 Orderly System Shutdown with SHUTDOWN.COM

```
$ @SYS$SYSTEM:SHUTDOWN

        SHUTDOWN---Perform an Orderly System Shutdown

How many minutes until final shutdown [0]: 10
Reason for shutdown: [Standalone] MONTHLY PREVENTIVE MAINTENANCE.
Do you want to spin down the disk volumes [No]? RET
Do you want to invoke the site-specific shutdown procedure [Yes]? RET
Should an automatic system boot be performed [No]? RET
When will the system be rebooted [later]? 12:30
Shutdown options:
 REBOOT_CHECK        Check existence of basic system files
Shutdown options [NONE] RET

%SHUTDOWN-I-OPERATOR, This terminal is now an operator's console.
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:04:00.15  %%%%%%%%%%%
Operator status for operator _AVALON$OPA0:
CENTRAL, PRINTER, TAPES, DISKS, DEVICES, CARDS, NETWORK, OPER1, OPER2,
OPER3, OPER4, OPER5, OPER6, OPER7, OPER8, OPER9, OPER10, OPER11,
OPER12

%SHUTDOWN-I-DISLOGINS, Interactive logins will now be disabled.
%SET-I-INTSET, login interactive limit = 0 current interactive value = 17
%SHUTDOWN-I-SHUTNET, The DECnet network will now be shut down.
%SHUTDOWN-I-STOPQUEMAN, The queue manager will now be stopped.

SHUTDOWN message on AVALON, from user SYSTEM at _AVALON$OPA0:   12:04:00.20
AVALON will shut down in 5 minutes; back up 12:30. Please log off node AVALON.
MONTHLY PREVENTIVE MAINTENANCE

17 terminals have been notified on AVALON.

SHUTDOWN message on AVALON from user SYSTEM at _AVALON$OPA0:   12:05:55.28
AVALON will shut down in 4 minutes; back up 12:30. Please log off node AVALON.
MONTHLY PREVENTIVE MAINTENANCE
```

(Continued on next page)

Example 4–1 (Cont.) Orderly System Shutdown with SHUTDOWN.COM

```
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:05:12.30  %%%%%%%%%%%
Message from user DECnet on AVALON
DECnet event 2.0, local node state change
From node 2.161 (AVALON), 16-JUN-1984 12:05:22.26
Operator command, Old state = On, New state = Shut
SHUTDOWN message on AVALON from user SYSTEM at _AVALON$OPA0:   12:07:12.56
AVALON will shut down in 3 minutes; back up 12:30. Please log off node AVALON.
MONTHLY PREVENTIVE MAINTENANCE
%SHUTDOWN-I-STOPQUEMAN, The queue manager will now be stopped.
SHUTDOWN message on AVALON user SYSTEM at _AVALON$OPA0:   12:07:12.56
AVALON will shut down in 3 minutes; back up 12:30. Please log off node AVALON.
MONTHLY PREVENTIVE MAINTENANCE
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:08:12:30  %%%%%%%%%%%
Message from user JOB_CONTROL on AVALON
-SYSTEM-S-NORMAL, normal successful completion
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:08:42.30  %%%%%%%%%%%
Message from user DECNET on AVALON
DECnet shutting down
SHUTDOWN message on AVALON from user SYSTEM at _AVALON$OPA0:   12:09:12.56
AVALON will shut down in 1 minute; back up 12:30. Please log off node AVALON.
MONTHLY PREVENTIVE MAINTENANCE

17 terminals have been notified on AVALON
%SHUTDOWN-I-SITESHUT, The site-specific shutdown procedure will now be invoked.
%SHUTDOWN-I-STOPUSER, All user processes will now be stopped.
%SHUTDOWN-I-REMOVE, All installed images will now be removed.
%SHUTDOWN-I-DISMOUNT, All volumes will now be dismounted.
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:09:42.30  %%%%%%%%%%%
Message from user System on AVALON
_AVALON$OPA0:, AVALON shutdown was requested by the operator.
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:10:02.44  %%%%%%%%%%%
Logfile was closed by operator _AVALON$OPA0:
Logfile was SYS$SYSROOT:[SYSMGR]OPERATOR.LOG;8
%%%%%%%%%%%  OPCOM, 16-JUN-1984 12:10:32.20  %%%%%%%%%%%
Operator _AVALON$OPA0: has been disabled, username SYSTEM
        SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
```

4.1.2 Emergency Shutdown with OPCCRASH

This section describes how to halt the system immediately without performing any of the housekeeping functions that ensure an orderly shutdown. Generally, you should shut down the system by following the orderly shutdown procedure described in Section 4.1.1, because since OPCCRASH performs only minimal housekeeping functions, data may be lost.

To perform this procedure, you must have the CMKRNL privilege. You can enter the commands from any terminal and any account.

1. Enter the following command to force an immediate shutdown of the system:

   ```
   $ RUN SYS$SYSTEM:OPCCRASH
   ```

2. If the system fails to respond, use the alternate crash procedure appropriate for your VAX processor, as described in Sections 4.1.3 and 4.1.4.

3. At the system console, observe the following message:

   ```
   SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
   ```

4. Now halt the system. You halt a VAX–11/730 system by pressing CTRL/P. To halt a VAX–11/780 system, press CTRL/P to obtain the console prompt (>>>), and then type HALT.

Example 4–2 illustrates an emergency shutdown on a VAX–11/780 using the OPCCRASH procedure.

Example 4–2 Emergency Shutdown on a VAX–11/780 Using OPCCRASH

```
$ RUN SYS$SYSTEM:OPCCRASH
     SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT SYSTEM
CTRL/P
>>>HALT
     HALTED AT 8000708A
```

4.1.3 Emergency Shutdown with CRASH on VAX–11/780 and VAX–11/730 Systems

This section describes the CRASH console program command file that forces VAX–11/780 and VAX–11/730 systems to fail, resulting in immediate shutdown. Procedures for performing the equivalent operation on VAX–11/750 systems are described in the next section.

Note: You should use CRASH only if the system will not accept command input; that is, if the system fails to accept or respond to OPCCRASH.

You must issue all commands that invoke CRASH from the system console terminal.

To force a VAX–11/780 or VAX–11/730 system to fail with CRASH, proceed as follows:

1. Halt the system. You halt a VAX–11/730 system by pressing CTRL/P. To halt all other VAX processors, press CTRL/P to obtain the console prompt (>>>), and then type HALT.

2. At the console prompt, type

   ```
   >>> @CRASH
   ```

   This command invokes the CRASH console program command file.

3. Additional messages and information, such as the fatal bugcheck message, are printed at the console terminal. The system dump file is written to the disk.

Example 4–3 illustrates the procedure for a VAX–11/780 system.

Example 4–3 Running CRASH on a VAX–11/780 System

```
CTRL/P
>>>HALT
>>>@CRASH
!
! Command file to crash VMS abnormally
!
HALT                 ! HALT SYSTEM, EXAMINE PC,
     HALTED AT 8000702A
EXAMINE PSL          ! PSL,
     00000000
EXAMINE/INTERN/NEXT:4 0 ! And all stack pointers
     I   00000000   80001D48
     I   00000001   00000000
     I   00000002   00000000
     I   00000003   00000000
     I   00000004   8009E600
DEPOSIT PC=-1           ! Invalidate PC
DEPOSIT PSL=1F0000      ! Kernel mode, IPL 31
CONTINUE
<@EOF>
<@EXIT>
```

(Continued on next page)

Example 4–3 (Cont.) Running CRASH on a VAX–11/780 System

```
**** FATAL BUG CHECK, VERSION = 4.0 INVEXCEPTN, Exception while above
ASTDEL or on interrupt stack
      CURRENT PROCESS = NULL
      REGISTER DUMP
         R0 = 0000001F
         R1 = 001F0000
         R2 = 00000000
         R3 = 00000000
         R4 = 00000000
         R5 = 00000000
         R6 = 00000000
         R7 = 00000000
         R8 = 00000000
         R9 = 00000000
         R10= 00000000
         R11= 00000000
         AP = 00000000
         FP = 00000000
         SP = 80001D14
         PC = 800038C1
         PSL= 001F0009
      KERNEL/INTERRUPT STACK
         80001D1C  00000004
         80001D20  00000000
         80001D24  FFFFFFFD
         80001D28  00000000
     .
     .
     .
      HALT INST EXECUTED
      HALTED AT 800071B3
      >>>
```

Typing @CRASH invokes the CRASH console program command file on the VAX–11/780 system. This procedure instructs the system to examine the program counter (PC), the processor status longword (PSL), and the stack pointers. Values are deposited in the PC and PSL to cause an exception condition that forces a system dump. The fatal bugcheck message is printed at the console terminal. Finally, the system halts and prints the contents of the program counter. The console system prompt (>>>) returns.

If automatic rebooting is enabled, some systems will halt and then automatically reboot. However, if it is not enabled, you must bootstrap the system manually.

In some cases, on VAX–11/782 attached processor systems only, there may be a delay of up to two minutes before the system responds to the @CRASH command.

4.1.4 Emergency Shutdown with CRASH on VAX–11/750 Systems

The console program command file CRASH is not included in the VAX/VMS distribution kit for a VAX–11/750 system. If it becomes necessary to perform an emergency crash of a VAX–11/750 system, you can enter the equivalent instructions manually, as shown in Example 4–4.

Example 4–4 Running CRASH on a VAX–11/750 System

```
CTRL/P
80007B06   02
>>>E/G F
      G        0000000F        80007B06
>>>E P
      00000000
>>>E/I 0
      I        00000000        800009D0
>>>E/I 1
      I        00000001        00000000
>>>E/I 2
      I        00000002        00000000
>>>E/I 3
      I        00000003        00000000
>>>E/I 4
      I        00000004        8013C000
>>>D/G F FFFFFFFF
>>>D P 1F0000
>>>>C
```

(Continued on next page)

Example 4–4 (Cont.) Running CRASH on a VAX–11/750 System

```
**** FATAL BUG CHECK, VERSION = 4.0 INVEXCEPTN, Exception while
above ASTDEL or on interrrupt stack
    CURRENT PROCESS = NULL
    REGISTER DUMP
   .
   .
   .
     HALT INST EXECUTED
     HALTED AT 800076B2
     >>>
```

The commands instruct the system to examine the program counter (PC), the processor status longword (PSL), and the stack pointers. Values are deposited in the PC and PSL to cause an exception condition that forces a system dump. The fatal bugcheck message is printed at the console terminal, as are additional messages and information. (See Example 4–3 for the VAX–11/780.) The system dump file is written to the disk. Later, the dump file can be used to determine why the system did not respond to command input. However, data may be lost, since no other housekeeping functions are performed. Finally, the system halts and prints the contents of the program counter. The console system prompt (> > >) returns.

4.2 Bootstrapping the System

You can bootstrap the system using either a default or an alternate bootstrap command procedure. In both cases, you must bootstrap the system directly from the system disk. If you are bootstrapping a VAX–11/750, you can use the stand-alone BOOT58 program on the console TU58 cartridge. This procedure is described in Appendix A.

Normally you bootstrap using the procedure you have selected as the default for your site (see Section 2.4). However, if the default procedure is unavailable, or if you are bootstrapping to create an environment for a specialized task (such as stand-alone debugging), you can select an alternate procedure. The next sections describe default and alternate bootstrap operations.

4.2.1 Default Bootstrap

The most direct way to bootstrap your operating system is to activate the **BOOT** switch or to enter console mode and type the BOOT command (or simply **B**). For example:

```
>>> B
```

In either case, the console program looks for your default bootstrap command procedure (DEFBOO.CMD) and executes it.

To bootstrap using the default, proceed as follows:

1 Halt the system. You halt a VAX–11/730 system by pressing CTRL/P. To halt all other VAX processors, press CTRL/P to obtain the console prompt (>>>), and then type HALT.

2 Either type BOOT or activate the **BOOT** switch on the processor control panel. On VAX–11/730 and VAX–11/780 systems, the default bootstrap command procedure DEFBOO.CMD is used to bootstrap the system. For VAX–11/750 systems, the processor will attempt to bootstrap from the device selected by the **BOOT DEVICE** switch on the front panel. If the switch is set to position A, the BOOT58 program is invoked (see Appendix A.)

4.2.2 Alternate Nonstop Bootstrap

You normally use an alternate nonstop bootstrap command procedure when the default procedure cannot be accessed because of problems with the device designated in the default procedure. To bootstrap the system using an alternate nonstop procedure, follow these steps:

1 Halt the processor. You halt a VAX–11/730 system by pressing CTRL/P. To halt all other VAX processors, press CTRL/P to obtain the console prompt (>>>), and then type HALT.

2 Bootstrap the system, using the command for your particular processor:

- **For the VAX–11/750**

```
>>> B ddcu
```

The code dd is the device type, c is the controller designator, and u is the unit number.

- **For all other VAX processors**

```
>>> B  ddu
```

The code dd is the device type, and u is the unit number.

4.2.3 Alternate Conversational Bootstrap

A conversational bootstrap is most commonly used in research and development environments to alter operating conditions for experimentation, testing, and debugging.This procedure allows the sophisticated user to

- Select an alternate file as the source of system parameter values
- Set and show individual parameter values
- Specify an alternate site-independent startup procedure
- Specify a minimum startup

To initiate a conversational bootstrap, proceed as follows:

1 Halt the processor. You halt a VAX–11/730 system by pressing CTRL/P. To halt all other VAX processors, press CTRL/P to obtain the console prompt (>>>), and then type the HALT command.

2 Bootstrap the system, using the command for your particular processor:

   - **For the VAX–11/750**

   ```
   >>> B/1 ddcu
   ```

   The code dd is the device type, c is the controller designator, and u is the unit number.

   - **For all other VAX processors**

   ```
   >>> @dduGEN
   ```

   The code dd is the device type, and u is the unit number.

3 When SYSBOOT is ready to accept commands, it prompts as follows:

```
%%
SYSBOOT>
```

The two percent signs that precede the SYSBOOT> prompt indicate a successful verification of the microcode. If the percent signs do not appear, discontinue the bootstrap operation and contact your field service representative.

When SYSBOOT prompts, you may use any of the subset of SYSGEN commands listed in Table 4–1 to examine or modify the system. You can find more information about these commands in the *VAX/VMS Utilities Reference Volume.*

Table 4–1 SYSGEN Commands Used in SYSBOOT

Command	Description
CONTINUE	Resumes the bootstrap procedure
DISABLE CHECKS	Inhibits checking of parameter values specified with the SET command
ENABLE CHECKS	Permits checking of parameter values specified with the SET command
HELP	Displays a summary of the SYSBOOT commands at your terminal
SET parameter-name	Establishes the value of a system generation parameter
SET/STARTUP	Sets the name of the system startup command procedure
SHOW [parameter]	Displays active, current, default, maximum, and minimum values for specific parameters; displays, with appropriate qualifiers, characteristics of parameters grouped by categories

Table 4–1 (Cont.) SYSGEN Commands Used in SYSBOOT

Command	Description
USE [file-spec]	Optionally specifies a parameter file to be used as a source of values (you must enter the entire file specification, including device and directory)

During a typical conversational bootstrap, you might enter the following commands to set a new value for the SYSGEN parameter WSMAX and continue the procedure:

```
SYSBOOT> SET WSMAX 512
SYSBOOT> CONTINUE
```

In this example, you set WSMAX to 512, and the bootstrap operation continues. When the VAX/VMS system announces itself, the new parameter value becomes active.

You can also use the conversational bootstrap operation to specify a minimum startup. For example, if you want to bootstrap your system and avoid autoconfiguring all your devices, issue the following command at the SYSBOOT> prompt:

```
SYSBOOT> SET STARTUP_P1 "MIN"
```

This command initiates a minimum startup that performs the following sequence of operations:

1. Starts the processes that control error logging, the job controller, and the operator's log
2. Installs known images
3. Defines the number of interactive users as eight
4. Logs off

Note that this minimum procedure does not invoke SYCONFIG.COM or SYSTARTUP.COM.

If you want verification set during the execution of a startup procedure, you can specify the following command:

```
SYSBOOT> SET STARTUP_P2 "YES"
```

The STARTUP_Pn parameters are not reset for the next bootstrap operation.

4.2.4 Bootstrapping an HSC Disk

The manner in which you bootstrap an HSC disk is in part dependent on the type of VAX processor you are using. If you have a VAX–11/750, you use the BOOT58 utility to bootstrap HSC disks (see Appendix A); VAX–11/780 users use console mode commands.

In either case, you must alter an appropriate bootstrap command file on your console volume: CIBOO.CMD if you want to use a nonstop boot, or CIGEN. if you want to use a conversational boot. If you are installing a new system; that is, not upgrading your current system, you must use CIBOO.CMD.

The CI bootstrap command procedures let you bootstrap from an HSC disk, but first you must add two pieces of information to the file: the unit number of the disk drive, and the node number of the HSC controlling it. Furthermore, if you want to bootstrap from a system root other than [SYS0], you must identify the root. Note, however, that when installing a system, you bootstrap from [SYS0]. See Section 2.8.2.2 for more information on bootstrapping from alternate system roots; see the *Guide to VAX/VMS Software Installation* for details on creating alternate roots.

There are two ways to provide the information needed by the CI bootstrap command procedures, depending on whether you have a running system or whether you are installing a new system.

If you have a running system, you can copy the CI bootstrap command file from the console volume to a disk directory, edit it, and then copy the edited version back to the console volume. If you want to use the CI bootstrap command file as your default bootstrap command procedure, copy the edited version to DEFBOO.CMD on the console volume.

If you are installing a new system, you will have to use console commands (BOOT58 commands if you have a VAX–11/750) to deposit the information needed by the CI bootstrap command files before bootstrapping your HSC device. Note that the form of the DEPOSIT command varies for different VAX processors.

The following sections describe the bootstrapping of HSC disks in four enviroments:

- On a VAX–11/780 during an installation
- On a running VAX–11/780
- On a VAX–11/750 during an installation
- On a running VAX–11/750

4.2.4.1 Bootstrapping an HSC Disk on a VAX–11/780 During an Installation

The following procedures assumes that you are bootstrapping an HSC disk on a VAX–11/780 while you are installing a new operating system.

1. Determine the unit number of the HSC disk and the node number of the HSC controlling it.
2. Press CTRL/P to put the system in console mode.
3. Use the console HALT command to stop the processor.
4. When the console prompt (>>>) appears, deposit the HSC node number into register R2 using the following command format.

   ```
   >>> DEPOSIT R2 node_number
   ```

   Note that all numeric entries are made using hexadecimal notation. For example, if the HSC is node number 12 on the CI780, you use this command:

   ```
   >>> DEPOSIT R2 0C
   ```

5. If the disk device is accessible to two HSCs, deposit both node numbers putting the greater number in hexadecimal digits 3 and 2, and the lesser in digits 1 and 0. For example, if one HSC is numbered 15 (hex 0E) and the other is numbered 10 (hex 0A), the proper command is

   ```
   >>> DEPOSIT R2 0E0A
   ```

6 Deposit the unit number of the HSC disk into register R3 using the following command format:

```
>>> DEPOSIT R3 unit_number
```

For example, if the HSC disk is unit number 21, deposit a hex 15 into R3:

```
>>> DEPOSIT R3 15
```

7 Bootstrap the HSC disk with this command:

```
>>> @CIBOO.CMD
```

4.2.4.2 Bootstrapping an HSC Disk on a Running VAX–11/780

The following procedure shows how to set up DEFBOO.CMD to do a nonstop system bootstrap of an HSC disk on a VAX–11/780 system that is up and running. For purposes of illustration, the HSC device is assumed to be an RA60 disk drive assigned unit number 03 on an HSC controller assigned node number 0C. The procedure assumes that you want to bootstrap the system from alternate root [SYSA].

1 Be sure your console drive is connected. If it is not, invoke the System Generation Utility (SYSGEN) to connect it, as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT CONSOLE
```

2 Exit from SYSGEN:

```
SYSGEN> EXIT
```

3 Mount the console volume:

```
$ MOUNT/FOREIGN CSA1:
```

4 Copy CIBOO.CMD to your default disk directory with this command:

```
$ EXCHANGE COPY CSA1:CIBOO.CMD *.*
```

5 Edit the disk file in your default directory as shown in the following example. The first line you add deposits the HSC node number in R2, and the second line you add deposits the HSC disk's unit number in R3.

To set the system root at [SYSA], you edit the line commented as DESIGNATED ROOT IS SYSA by changing the most significant digit in R5 to the letter A. This digit determines the system root for system disks with multiple systems.

Note that all numeric entries are made using hexadecimal notation.

```
    .
    .
    .
DEPOSIT R0 20               !CI PORT DEVICE
DEPOSIT R1 E                !CI TR=E
DEPOSIT R2 0C               !HSC NODE NUMBER
DEPOSIT R3 03               !DEVICE UNIT NUMBER
DEPOSIT R4 0                !BOOT BLOCK LBN (NOT USED)
DEPOSIT R5 A0004000         !DESIGNATED ROOT IS SYSA
    .
    .
    .
```

6 If the disk is dual-ported (accessible to two HSC controllers), you must specify the node number of both controllers in register R2. Further, you must specify the higher-numbered node in bits <15:8> and the lower-numbered node in bits <7:0>. For example, if the disk is accessible to the HSC numbered *0C* and the HSC numbered *0A*, the line for R2 should appear like this:

```
DEPOSIT R2 0C0A          !DUAL-PORTED HSC NODE NUMBERS
```

7 Copy the edited bootstrap command file to the default bootstrap command procedure (DEFBOO.CMD) on your console volume with this command:

```
$ EXCHANGE COPY device:[directory]CIBOO.CMD CSA1:DEFBOO.CMD
```

You should provide a spare copy of this file in case DEFBOO.CMD is overwritten. Use the same command, but provide a distinctive filename for the spare file.

8 Shut down the system with this command:

```
$ @SYS$SYSTEM:SHUTDOWN
```

9 Press CTRL/P to put the system into console mode.

10 Use the HALT command to stop the processor.

11 Bootstrap the HSC disk by invoking the default bootstrap command procedure:

```
>>> B
```

4.2.4.3 Bootstrapping an HSC Disk on a VAX–11/750 During Installation

The following procedure shows how to bootstrap an HSC disk on a VAX–11/750 while installing a new operating system.

1. At the processor control panel, set the **BOOT SELECT** switch to position A.

2. Determine the unit number of the HSC disk and the node number of the HSC controlling it.

3. Press CTRL/P to put the system in console mode.

4. When the console prompt (>>>) appears, invoke BOOT58 with this command:

```
>>> B/800 DDA0
```

Note: If you fail to add the /800 qualifier, the console program will try to execute DEFBOO.CMD.

5. When the BOOT58> prompt appears, deposit the HSC node number into register R2 using the following command format:

```
BOOT58> D/G 2 node_number
```

Note that all numeric entries are made using hexadecimal notation. For example, if the HSC is node number 12 on the CI750, you use this command:

```
BOOT58> D/G 2 0C
```

6. If the load device is accessible to two HSCs, deposit both node numbers, putting the greater number in hexadecimal digits 3 and 2, and the lesser in digits 1 and 0. For example, if one HSC is numbered 15 (hex 0E) and the other is numbered 10 (hex 0A), the proper command is

```
BOOT58> D/G 2 0E0A
```

7 Deposit the unit number of the load device into register R3 using the following command format:

```
BOOT58> D/G 3 unit_number
```

For example, if the load device is unit number 21, deposit a hex 15 into R3:

```
BOOT58> D/G 3 15
```

8 Bootstrap the HSC disk with this command:

```
BOOT58> @CIBOO.CMD
```

4.2.4.4 Bootstrapping an HSC Disk on a Running VAX–11/750

This section describes how to set up DEFBOO.CMD to do a nonstop system bootstrap of an HSC disk on a VAX–11/750 system that is up and running. For purposes of illustration, the HSC device is assumed to be an RA60 disk drive assigned unit number 03 on an HSC controller assigned node number 12 (hexadecimal 0C). It is further assumed that the user wants to bootstrap the system from alternate root SYSA.

1 Set the **BOOT SELECT** switch to position A.

2 Be sure your console drive is connected. If it is not, invoke the System Generation Utility (SYSGEN) to connect it, as follows:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN> CONNECT CONSOLE
```

3 Exit from SYSGEN:

```
SYSGEN> EXIT
```

4 Mount the console volume:

```
$ MOUNT/FOREIGN CSA1:
```

5 Use the Exchange Utility (EXCHANGE) to copy CIBOO.CMD to your default disk directory.

```
$ EXCHANGE COPY CSA1:CIBOO.CMD *.*
```

6 Edit the disk file in your default directory and add the two lines commented as "HSC NODE NUMBER" and "DEVICE UNIT NUMBER" . The first line deposits the HSC node

number in register R2, and the second line deposits the HSC disk's unit number in register R3.

To set the system root at SYSA, you edit the line commented as SOFTWARE BOOT CONTROL FLAGS by changing the most significant digit in register R5 to the letter A. This digit determines the system root for system disks with multiple systems.

Note that all numeric entries are made using hexadecimal notation.

```
    .
    .
    .
D/G R0 20                !CI PORT DEVICE
D/G R1 E                 !CI TR=E
D/G R2 0C                !HSC NODE NUMBER
D/G R3 03                !DEVICE UNIT NUMBER
D/G R4 0                 !BOOT BLOCK LBN (NOT USED)
D/G R5 A0004000          !SOFTWARE BOOT CONTROL FLAGS
    .
    .
    .
```

7 If the disk is dualported (accessible to two HSC controllers), you must specify the node number of both controllers in register R2. Further, you must specify the higher-numbered node in bits <15:8> and the lower-numbered node in bits <7:0>. For example, if the disk is accessible to the HSC numbered 12 (hexadecimal 0C) and the HSC numbered 10 (hexadecimal 0A), the line for register R2 should appear like this:

```
D/G R2 0C0A              !DUAL-PORTED HSC NODE NUMBERS
```

8 Copy the edited boot command to the default boot command procedure (DEFBOO.CMD) on your console volume.

```
$ EXCHANGE COPY device:[directory]CIBOO.CMD CSA1:DEFBOO.CMD
```

You should provide another copy of this file in case DEFBOO.CMD is overwritten. Use the same command, but provide a distinctive file name for the other file.

9 Write the boot block on the TU58 using the Writeboot Utility.

   a Invoke WRITEBOOT:

```
$ RUN SYS$SYSTEM:WRITEBOOT
```

   b When WRITEBOOT asks for the target device, make the following entry:

```
Target system device (and boot file if not VMB.EXE):? CSA1:BOOT58.EXE
```

   c In response to the following question, enter the number 1:

```
Enter VBN of boot file code (default is one): 1
```

   d When you are asked for the address of the primary bootstrap, enter C000:

```
Enter load address of primary bootstrap in HEX (default is 200): C000
```

10 Shut down the system:

```
$ @SYS$SYSTEM:SHUTDOWN
```

11 Press CTRL/P to put the system in console mode:

12 When the console prompt appears (>>>), bootstrap the HSC disk:

```
>>> B
```

4.2.5 Bootstrapping a VAX–11/780 with Interleaved Memory

To bootstrap a VAX–11/780 system with interleaved memory, you must verify that the system conforms to certain requirements, as described in the *VAX Hardware Handbook*. If your system meets these requirements, edit the default bootstrap command procedure (DEFBOO.CMD) and the powerfail restart command procedure (RESTAR.CMD) to include commands that modify the memory controller registers.

DEFBOO.CMD and RESTAR.CMD are on the console floppy. Copy the files from the console floppy, edit them, and return them to the floppy using the DXCOPY command procedure described in Section 2.5.

4.3 Restarting Problems

Sometimes the operating system does not bootstrap after you have issued the BOOT command. This can be caused by either a hardware or software malfunction.

4.3.1 Hardware Problems

A read error on a disk drive or console medium, or a machine check error may indicate a hardware malfunction. When a hardware problem occurs, a question mark (?) usually precedes the error message that is displayed on the system console terminal. You should then:

1. Consult the appropriate hardware manual for your VAX processor.

2. Contact the appropriate DIGITAL Field Service representative.

4.3.2 Software Problems

When the operating system is loaded into memory but the STARTUP.COM command procedure does not execute, a software malfunction has probably occurred. You should suspect this condition if the usual message specifying the number of interactive users does not appear.

You can perform one or both of the following actions to correct the situation:

- Try again, by repeating the bootstrap procedure to restart (see the software installation booklet for your VAX processor).
- Place the system disk in another drive or try a different copy of the disk in the same drive, and repeat the restarting procedure.

5 Managing System Users

Effective system management depends to a large degree on effective user management. The control of system functions on a user-by-user basis plays a vital role in managing system resources and thus enhancing system efficiency and security.

In order to manage users effectively, you must understand

- Who needs access to the system
- What tasks they must perform
- What system resources they will require
- What information you should provide them

Once you understand user needs, you can establish controls that will tailor the system appropriately. VAX/VMS provides several tools to authorize and control the use of system resources by individual users. This chapter describes these tools and discusses how and when to use them.

5.1 User Authorization File

In VAX/VMS, user management is accomplished primarily through user accounts, which control who may log in to the system and how individuals may use it. By creating and maintaining user accounts, you control how your users operate the system.

User accounts are defined by records in a file (SYS$SYSTEM:SYSUAF.DAT) called the user authorization file (UAF), which you maintain using the Authorize Utility (AUTHORIZE). A detailed explanation of how to use AUTHORIZE is presented in the *VAX/VMS Utilities Reference Volume.*

Whenever a user logs in, the system uses the information contained in the UAF to validate the login attempt, establish the

account's environment, and create a process with appropriate attributes. In this way, the system restricts users to the resources you assign to each account.

The procedures for adding a user account are discussed in detail later in this chapter. Since, however, the UAF is the prime repository for storing information about user accounts, it is important to understand its components before you add accounts.

Using the Authorize Utility, you create and maintain UAF records by assigning values to various *fields* within each record. The values you assign identify the user, define his or her working environment, and control use of system resources. Example 5–1 presents a typical UAF record for a restricted user account and describes its fields.

Example 5–1 Sample UAF record

```
Username: WELCH                              Owner:  ROB WELCH       ❶
Account:                                     UIC:    [21,51] ([INV,WELCH])
CLI:      DCL                                Tables: DCLTABLES       ❷
Default:  USER3:[WELCH]
LGICMD:
Login Flags:  Diswelcome Disnewmail                                  ❸
Primary days:    Mon Tue Wed Thu Fri
Secondary days:                     Sat Sun
Primary   000000000011111111112222  Secondary 000000000011111111112222
Day Hours 012345678901234567890123  Day Hours 012345678901234567890123
Network:  ------ No access -------            ----- Full access ------
Batch:    XXXXXXXXX--------XXXXXXX            ---------XXXXXXXXX------
Local:    XXXXXXXXX--------XXXXXXX            ---------XXXXXXXXX------
Dialup:   ----- Full access ------            ------ No access -------
Remote:   ----- Full access ------            ------ No access -------
Expiration:           (none)   Pwdminimum:  6   Login Fails:     0
Pwdlifetime:          (none)   Pwdchange:   15-APR-1984 13:58
Last Login:           (none) (interactive),           (none) (non-interactive)
Maxjobs:          0 Fillm:        20 Bytlm:          8192            ❹
Maxacctjobs:      0 Shrfillm:      0 Pbytlm:            0
Maxdetach:        0 BIOlm:        18 JTquota:        1024
Prclm:            2 DIOlm:        18 WSdef:           150
Prio:             4 ASTlm:        24 WSquo:           200
Queprio:          4 TQElm:        10 WSextent:        500
CPU:         (none) Enqlm:        30 Pgflquo:       10000
Authorized Privileges:                                               ❺
  TMPMBX NETMBX
Default Privileges:
  TMPMBX NETMBX
```

❶ **User identification fields** contain information used by the system for both accounting purposes and user identification.

❷ **Default fields** contain the default specifications for

- The command language interpreter (DCL by default)
- The name of the command procedure to be executed automatically at login time. If the field is blank, SYS$LOGIN:LOGIN.COM is executed by default.
- The command language interpreter tables (if blank, same as CLI)
- The device and directory names for file access

❸ **Login characteristics fields** impose specific login restrictions that

- Inhibit certain login functions
- Control the days of the week when various types of logins are permitted
- Control the time of day when various types of logins are permitted

❹ **Resource control fields** control system resources by

- Limiting the use of reusable system resources
- Specifying the base priority used in scheduling the process that the system creates for the user

❺ **Privileges fields** specify the privileges that allow use of restricted and sensitive system functions.

5.1.1 System-Supplied UAF Records

The software distribution kit provided with a new VAX/VMS system contains a UAF of four records:

- **DEFAULT**—Serves as a template for creating user records in the UAF. A new user record is assigned the values of the DEFAULT record except where you explicitly override those values. Thus, whenever you add a new account, you need only specify values for fields that you want to be different. For example, the following AUTHORIZE command creates a new record having the same values as the DEFAULT record, except that the password, UIC, and default directory fields are changed.

  ```
  UAF> ADD MARCONI/PASSWORD=QLP6YT9A/UIC=[033,004] -
  _/DIRECTORY=[MARCONI]
  ```

  Section 5.3 gives an example of how to use AUTHORIZE to add a user account.

  You can also use AUTHORIZE to modify the DEFAULT record to suit your needs or to create additional default record templates for specific account types. Section 5.5.1 explains how to create and use additional default templates.

Note: The default record cannot be renamed or deleted from the UAF.

- **FIELD**—Permits DIGITAL Field Service personnel to check out a new system. The FIELD record can be deleted once the system is installed.
- **SYSTEM**—Provides a means for you to log in with full privilege. The SYSTEM record can be modified but cannot be renamed or deleted from the UAF. Note that if you change the SYSTEM record, particularly the default device and directory and the privileges, you could prevent successful installation of optional software products or future VAX/VMS maintenance releases.
- **SYSTEST**—Provides an appropriate environment for running the User Environment Test Package (UETP). The SYSTEST record can be deleted once the system is installed.

5.1.2 General Maintenance of the UAF

Typically, you use the UAF supplied with the distribution kit. (You can, however, rename the UAF with the DCL command RENAME, and then create a new UAF with AUTHORIZE.) You should limit any kind of access to this file to the SYSTEM account (see the *Guide to VAX/VMS System Security* for guidelines on protecting system files). Furthermore, each time you modify the file, you should create a backup copy so that in the case of a system failure you do not lose the modifications. See the *Guide to VAX/VMS Disk and Magnetic Tape Operations* for guidelines on backing up files.

The UAF is accessed as a shared file, and updates to the UAF are made on a per-record basis, which eliminates the need for both a temporary UAF and a new version of the UAF after each AUTHORIZE session. Updates become effective as soon as AUTHORIZE commands are entered, not after the termination of AUTHORIZE. (For this reason, you should not enter temporary values with the intent of fixing them later in the session.)

Immediately after installing the system, you should make the following modifications to the UAF:

- **SYSTEM, FIELD, and SYSTEST accounts**—Change the passwords immediately. Use obscure passwords of six characters or more and continue to change them on a regular basis. These accounts permit access to the system that the general user should not have. As an alternative to changing the password, you could specify MODIFY /FLAGS=DISUSER with AUTHORIZE, especially if you use the account infrequently. The login flag DISUSER disables the account and prevents anyone from logging in on the account. To enable the account when it is needed, run AUTHORIZE and specify MODIFY/FLAGS=NODISUSER.

  As a general rule, do not make any other changes in the SYSTEM UAF record; installation of VAX/VMS maintenance releases and optional software products depends on certain values in this record.

- **DEFAULT account**—You may want to change several fields in this account. For example:

  ```
  UAF>MODIFY DEFAULT/DEVICE=DISK$USER/PGFLQUOTA=25000
  ```

The default device is set to the name most commonly used for user accounts that will be added. Likewise the page file quota value is set to a typical value for most users at the site.

Note that you use the SYSTEM account only for system functions such as performing backups and installing maintenance updates. The account comes to you with full privileges, so you must exercise caution in using it. For example, because you have BYPASS privilege, the system will allow you to delete any file no matter what its protection. If you type an incorrect name or spurious asterisk, you may destroy files that you or other users need to keep. For this reason, you should use another account with fewer privileges for day-to-day system management activities.

If you want to receive mail on the SYSTEM account, you can define a system-wide logical name in the site-specific startup command procedure (see Chapter 2) to equate SYSTEM to the user name of the system manager's account. You can also use the MAIL command SET FORWARD to have any SYSTEM mail forwarded to any other account.

5.2 Preparing to Add a User Account

How you set up a user account depends on the needs of the individual user. In general, there are two types of accounts:

- **Interactive**—A person using an interactive account has access to the system software and can perform work of a general nature (program development, text editing, and so on). Normally, such an account is considered individual; that is, only one person can use it.

- **Turnkey**—A person using a turnkey or captive account has access only to user software and can only perform strictly limited work. Normally, such an account is considered functional; that is, anyone who needs to perform the particular work can use it. As an example, you might develop an inventory system. Anyone whose job entails inventory control can access your system, but that person cannot access other systems or the base software.

The suggested procedures for preparing to add a user account are as follows:

1 Determine a user name and password.

2 Determine a user identification code (UIC).

3 Use the Disk Quota Utility (DISKQUOTA) to add a disk quota entry for this UIC, if disk quotas are in effect.

4 Decide where the account's files will reside (which device and directory) and create a first-level directory on the appropriate volume.

5 Establish any login/logout command files.

When you have completed these steps, you can invoke the Authorize Utility as described in Section 5.3 and add the account.

The sections that follow discuss each step in the creation of a user account and introduce the various management utilities and commands used. Detailed descriptions of the utilities and commands appear in the *VAX/VMS Utilities Reference Volume* and the *VAX/VMS DCL Dictionary*, respectively.

5.2.1 User Name and Password

To determine a user name and password, use naming conventions that take into consideration the nature of the account. For example, interactive accounts are usually named after the person using the account. Turnkey accounts, on the other hand, use a name that describes the function of the account. Thus, an interactive account for Robert Jones would typically have a user name of JONES, while a turnkey account for an inventory system would typically be called INVENTORY. On systems with a large number of users, you should remember to assign user names that are unique.

For interactive accounts, it is best to let the person using the account control the password. Initially, you provide a simple password and tell the user to change the password with the DCL command SET PASSWORD. Only the person using the account need know the password. You should encourage all

users to set obscure passwords of at least six characters and to change them frequently.

For turnkey accounts, the degree of sensitivity of the data used by the account should determine the type of password. For example, the password for a payroll application should be obscure, while the password for a suggestions account might not even be required; it could be null (in which case all one would need to do to log in is press RETURN at the PASSWORD: prompt). For all turnkey accounts, prohibit users from changing the password. To do this, specify /FLAGS=LOCKPWD when you add the account with the Authorize Utility. Change the password whenever you feel it might be compromised (for example, if a person using the account moves to another job).

5.2.2 User Identification Code

See the *VAX/VMS DCL Dictionary* for a detailed discussion of the UIC. In general, you assign each account a unique UIC, and you should assign accounts the same group number if they perform similar work, access the same files frequently, and/or use many of the same logical names.

5.2.3 Disk Quota Entry

If disk quotas are in effect for the volume, run the Disk Quota Utility (DISKQUOTA) to add an entry for the new UIC. Disk quotas limit the amount of disk space available to individual users on a particular volume. To add a disk quota entry, run DISKQUOTA as follows:

```
$ RUN SYS$SYSTEM:DISKQUOTA
DISKQ> USE DISK$USER
DISKQ> ADD [014,006] /PERMQUOTA=2000/OVERDRAFT=500
DISKQ> EXIT
```

The commands in this example

- Invoke DISKQUOTA
- Specify the disk DISK$USER as the volume on which quotas are set
- Add a disk quota entry of 2000 blocks for UIC [014,006] and assign the entry an overdraft of 500 blocks

- Exit from the utility

The sum of the quota and overdraft values is the absolute maximum number of blocks allotted to the user, which in this example is 2500 blocks. A discussion of disk quotas is presented in Section 7.8.2.1. The Disk Quota Utility is described in detail in the *VAX/VMS Utilities Reference Volume*.

5.2.4 User Directory and Default File Specification

For each interactive account, you should create a first-level directory (using the DCL command CREATE/DIRECTORY) under which the interactive user can create and maintain files and subdirectories. Make the owner of the directory the UIC you have decided upon for the new account. Typically, you also use the name of the account for the first-level directory. For example, if you have decided upon an account name of JONES and a UIC of [014,006], you would issue the following DCL command to create a first-level directory for the account on the volume DISK$USER:

```
$ CREATE/DIRECTORY DISK$USER:[JONES]/OWNER_UIC=[014,006]
```

The volume on which the directory is established depends on which devices you reserve for interactive accounts and how much space is available on each.

The default file specification you provide the new account (when you run AUTHORIZE) should be the name of the device and the name of the first-level directory you used in the DCL command CREATE/DIRECTORY.

For a turnkey account, whether you create a first-level directory depends on the nature of the user system. Where the user system uses files in a particular directory, you should make that directory the default directory specification. For example, if the inventory system uses the files DISK$DATA:[INV]STOCK1.DAT and DISK$DATA:[INV]STOCK2.DAT, the default device specification should be DISK$DATA: and the default directory specification should be [INV].

5.2.5 Login Command Procedures for Interactive Accounts

For interactive accounts, login command procedures contain commands commonly executed at the beginning of every user session. These commands do such things as

- Define symbols
- Assign logical names
- Display messages and the time of day
- Set terminal characteristics
- Define keys to perform certain functions

Login command procedures are useful for both saving keystrokes and standardizing operations. In establishing login command procedures for interactive accounts, you have the following choices:

- **System**—As system manager, you normally create and maintain a standard login command procedure in the system directory (the file is typically named SYS$MANAGER:SYLOGIN.COM). You then equate the logical name SYS$SYLOGIN to the name of the file so that whenever a user logs in, the procedure is executed.
- **Individual**—For any or all accounts, you may specify an additional login command procedure with the /LGICMD qualifier of AUTHORIZE. You can give the login command file any valid file specification. Then, whenever the user logs in, the specified procedure is executed after SYLOGIN.COM.
- **User-specified command file**—If individual or system login command procedures are not implemented, the system looks for a command file called LOGIN. in the user's login directory (as defined by the SYS$LOGIN logical name). If the file is found, the system executes it. This command file is developed and maintained by the user and should follow these conventions:
  - Device and directory names must take the default file specification for the account.
  - The filename must be LOGIN.

As an aid to new users, you might copy a login command procedure template into newly created first-level directories. However, to ensure proper ownership of the file, you must change the owner UIC of the file to that of the user. You make this change with the DCL command SET FILE /OWNER_UIC.

Examples 5–2 and 5–3 illustrate typical system and user-specified login command procedures.

As Example 5–2 shows, you can disable the CTRL/Y function (which suspends execution of the current image and invokes the command interpreter) to force execution of the complete login command procedure whenever the user logs in. You can do this with the DCL command SET NOCONTROL=Y. The login command procedure should, at some point, reset the CTRL/Y function with the DCL command SET CONTROL=Y.

Example 5–2 Sample SYS$MANAGER:SYLOGIN.COM Login Command Procedure

```
$ V = F$VERIFY(0)
$START:
$ !
$ SET NOCONTROL=Y  ! Do not allow CTRL/Y to exit procedure
$ SET NOON
$ !
$ !     Set default file protection back to the old default
$ !
$ SET PROTECTION=(SY:RWED,OW:RWED,GR:RWED,WO:RE)/DEFAULT
$ !
$ !     Allow network jobs to start faster
$ !
$ IF F$MODE() .EQS. "NETWORK" THEN GOTO EXIT
$ !
$ !     Enable CTRL/T handling by DCL
$ !
$ SET CONTROL=T
$ !
$ !     Define Foreign Commands For Installed Utilities
$ !
$ SDA                ==    "ANALYZE/CRASH_DUMP"
$ USERS              ==    "SHOW USERS"
$ DISPLAY            ==    "MONITOR PROCESSES/TOPCPU"
$ NCP                ==    "$NCP"
$ INFO               ==    "SHOW PROCESS/CONTINUOUS"
$ SUSPEND            ==    "SET PROCESS/SUSPEND"
$ RESUME             ==    "SET PROCESS/RESUME"
$ SETNAME            ==    "SET PROCESS/NAME"
$ !
$ !     Define a symbol indicating whether the terminal
$ !     is on a dialup port
$ !
$ TT == F$GETDVI("TT","DEVNAM")-"-"
$ DIALUP == ((TT .GES. "TTG0:" .AND. TT .LES. "TTG4:") -
        .OR. (TT .GES. "TTH1:" .AND. TT .LES. "TTH4:") -
        .OR. (TT .EQS. "TTI5:"))
$ IF DIALUP THEN SET TERMINAL/INQUIRE
$ !
$EXIT:
$ IF V THEN SET VERIFY
    .
    .
    .
$ SET CONTROL=Y
$ EXIT
```

Example 5–3 Sample LOGIN.COM Login Command Procedure

```
$ SET NOON
$ SET PROTECTION=(S=RD,O=RWED,G=R,W=R)/DEFAULT
$ !
$ ! Define abbreviations for often used commands
$ !
$ DIR*ECTORY    ==      DIRECTORY/DATE/SIZE
$ PH*ONE        ==      PHONE/SCROLL
$ !
$ !
$ ! Other useful abbreviations
$ !
$ SHP           ==      "SHOW PROCESS/PRIVILEGES"
$ PRI*NT        ==      "PRINT/NOTIFY"
$ SHD           ==      "SHOW DEFAULT"
$ UP            ==      "SET DEFAULT [-]"
$ SP            ==      "SET PROCESS/PRIVILEGES="
$ SQ            ==      "SHOW QUEUE/BATCH/ALL/DEVICE"
$ H*OME         ==      "SET DEFAULT SYS$LOGIN"
$ SUB*MIT       ==      "SUBMIT/NOTIFY"
$ SPC           ==      "SHOW PROCESS/CONTINUOUS"
$ SYS           ==      "SHOW SYSTEM"
$ DAY           ==      "SHOW TIME"
$ !
$ ! Set /LOG for all commands
$ !
$ BACK*UP       ==      "BACKUP/LOG"
$ DEL*ETE       ==      "DELETE/LOG"
$ LIB*RARY      ==      "LIBRARY/LOG"
$ PUR*GE        ==      "PURGE/LOG"
$ REN*AME       ==      "RENAME/LOG"
$ !
$ ! End of LOGIN.COM processing
$ !
$ GOTO F$MODE()
$NETWORK:
$ EXIT
$INTERACTIVE:
$ VN            ==      "SET TERMINAL/WIDTH=80"
$ VW            ==      "SET TERMINAL/WIDTH=132"
$ EXPERT        ==      "SET MESSAGE/NOFACIL/NOSEVER/NOIDENT"
$ NOVICE        ==      "SET MESSAGE/FACILITY/SEVERITY/IDENTIF"
$ NOVICE
```

(Continued on next page)

Example 5–3 (Cont.) Sample LOGIN.COM Login Command Procedure

```
$ !
$ ! Symbols for network users
$ !
$ SYSA          ==      "SET HOST SYSA"
$ SYSB          ==      "SET HOST SYSB"
$ SYSC          ==      "SET HOST SYSC"
$ EXIT                              ! End of interactive login
$BATCH:
$ SET VERIFY                        ! End of batch login
$ EXIT
```

5.2.6 Login Command Procedures for Turnkey Accounts

For turnkey accounts, the login command procedure specified by the /LGICMD qualifier of AUTHORIZE directs the account user into an application program and logs the user out upon termination of the task. The login command procedure for the inventory system, for example, might consist of the following commands:

```
$ DEFINE SYS$DISK DISK$INVENT
$ RUN INVENTORY
$ LOGOUTNOW
```

The application program INVENTORY assumes control transparently to the person logging into the account. You should assign the CAPTIVE flag to the login flags field of the turnkey account record by specifying the AUTHORIZE qualifier /FLAGS=CAPTIVE. The CAPTIVE flag locks the user of the turnkey account into the application software. Section 5.3 shows how to use AUTHORIZE to create a UAF record for a turnkey account.

5.2.7 Logout Command Procedures

The system does not provide for automatic execution of a command procedure at logout time. However, you can supply one as follows:

1. Create a system-wide logout command procedure that executes whenever a user logs out (the file is typically named SYS$MANAGER:SYLOGOUT.COM).

2. To ensure that this command procedure always executes, include a command in SYS$MANAGER:SYLOGIN.COM that equates the most commonly used abbreviation of the LOGOUT command (often LO) to the execution of the logout command procedure. For example:

```
$ LO*GOUT:==@SYS$MANAGER:SYLOGOUT
```

   The last line of the logout command procedure then uses an alternate form of the LOGOUT command, such as a LOGOUTNOW command. (You can create any command name you like beginning with LO.) You cannot use the same abbreviation as used for the symbol (in this case LO) because it will start the procedure again. As an alternative, you could make the next to the last line of the procedure:

```
$ DELETE/SYMBOL/GLOBAL LOGOUT
```

5.3 Using AUTHORIZE to Add a User Account

Once you have analyzed the purpose of a user account and decided which attributes and resources it requires, you can use the Authorize Utility to create it.

Give yourself the SYSPRV privilege. Then issue the following commands to set your default device and directory to that of SYS$SYSTEM and invoke the utility:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF>
```

When the utility responds with the UAF> prompt, you can add a typical interactive account as follows:

```
UAF> ADD JONES/PASSWORD=LPB57WM/UIC=[014,006] -
_/DEVICE=DISK$USER/DIRECTORY=[JONES] -
_/LGICMD=DISK$USER:[NEWPROD]GRPLOGIN -
_/OWNER="ROBERT JONES"/ACCOUNT=NEWPROD
```

The /OWNER and /ACCOUNT specifications are primarily for accounting purposes and can be omitted. The following unspecified qualifiers typically take their default values from the DEFAULT record:

- **Limits and Quotas** (/ASTLM, /BIOLM, /CPUTIME, /DIOLM, /ENQLM, /FILLM, /JTQUOTA, /MAXACCTJOBS, /MAXDETACH, /MAXJOBS, /PGFLQUOTA, /PRCLM, /SHRFILLM, /TQELM, /WSDEFAULT, /WSEXTENT, /WSQUOTA)—These qualifiers impose limits on the use of reusable system resources. See Chapter 6 for a discussion of limits and quotas; the defaults are normally adequate.
- **Priority** (/PRIORITY, /QUEPRIORITY)—The default values are normally adequate for accounts not running real-time processes. See Chapter 6 for a discussion of the processor priorities.
- **Privileges** (/DEFPRIVILEGES, /PRIVILEGES)—See Chapter 6 for a discussion of privileges; the default privileges (TMPMBX, NETMBX) are normally adequate.
- **Primary and Secondary Login Times; Login Functions** (/ACCESS, /DIALUP, /FLAGS, /INTERACTIVE, /LOCAL, /PRIMEDAYS, /REMOTE)—By default, users are allowed to log in at any hour of any day. To override the setting of a particular day, you can use the DCL command SET DAY. Use this command if a holiday occurs on a day that would normally be treated as a primary day and you want it treated as a secondary day. See Section 5.5.4 for a discussion of using these fields to restrict login times and functions.
- **Command Language Interpreter (DCL)**—Specify MCR if running in RSX compatibility mode.

The following example shows an AUTHORIZE command that adds the record for a turnkey account:

```
UAF> ADD INVENTORY/PASSWORD=QRC7Y94A/UIC=[033,066] -
_/DEVICE=DISK$INVENT/DIRECTORY=[INV]/LGICMD=INVENTORY -
_/FLAGS=CAPTIVE/NOACCESS=(PRIMARY, 18-8, SECONDARY, 0-23)
```

In this example, the /FLAGS and /NOACCESS qualifiers are used to restrict users from logging in to the turnkey account. The /NOACCESS qualifier limits logins to specific hours (see Section 5.5.4). The /FLAGS=CAPTIVE qualifier adds the login flag CAPTIVE to the turnkey account record. The CAPTIVE flag locks the person using the account into the application software by

- Disabling the CTRL/Y function to prevent users from interrupting the execution of the command procedure and gaining access to the command interpreter
- Preventing the user from specifying an alternate command interpreter with the /CLI qualifier at login time
- Preventing the user from specifying an alternate default disk device with the /DISK qualifier at login time

You may also want to disable mail and welcome notices with the /FLAGS=(DISNEWMAIL,DISWELCOME) qualifier.

5.4 Using a Command Procedure to Add a User Account

For consistent results, you can incorporate the steps for adding a user account (interactive or turnkey) into a command procedure similar to that shown in Example 5–4.

Example 5–4 Command Procedure for Adding a User

```
$ !
$ !   ADDUSER.COM---Adds a new user to the system authorization file
$ !
$ USERDISK = "WRKD$:"               ! Default disk for new users
$ UAF = "$AUTHORIZE"
$ ON CONTROLY THEN GOTO CLEANUP
$ ON WARNING THEN GOTO CLEANUP
$ OLDDIR = F$ENVIRONMENT("DEFAULT")
$ PREVPRIV = F$SETPRV("SYSPRV")
$ IF .NOT. F$PRIVILEGE("SYSPRV") THEN GOTO NOPRIV
$ SET DEFAULT SYS$SYSTEM
$ !
$ ! Request account information
$ !
$ INQUIRE USERNAME "Username"
$ INQUIRE FULLNAME "Full name"
$ SET TERMINAL/NOECHO
$ INQUIRE PASSWORD "Password [''Username']"
$ SET TERMINAL/ECHO
$ IF PASSWORD .EQS. "" THEN PASSWORD = USERNAME
$GET_GRP:
$ INQUIRE GRP "UIC Group Number"
$ IF GRP .EQS. "" THEN GRP = "*"
$ WRITE SYS$OUTPUT ""
$ WRITE SYS$OUTPUT "Determine the UIC from the following listing:"
$ WRITE SYS$OUTPUT ""
$ UAF SHOW ['GRP',*]/BRIEF
$ INQUIRE UIC
$ IF UIC .EQS. "" THEN GOTO GET_GRP
$ IF F$LOCATE("[",UIC) .EQ. F$LENGTH(UIC) .AND. -
     F$LOCATE("<",UIC) .EQ. F$LENGTH(UIC) THEN UIC = "[" + UIC + "]"
$ INQUIRE ACCOUNT "Account Name [VMS]"
$ IF ACCOUNT .EQS. "" THEN ACCOUNT = "VMS"
$ INQUIRE PRIVS "Privileges [NONE]"
$ IF PRIVS .NES. "" THEN PRIVS = "/PRIV=(" + PRIVS + ")"
$ USERDIR = F$EXTRACT(0,9,USERNAME)
$ INQUIRE TMP "Login Directory [''USERDIR']"
$ IF TMP .NES. "" THEN USERDIR = TMP
$ INQUIRE TMP "Login Device [''USERDISK']"
$ IF TMP .NES. "" THEN USERDISK = TMP
$ DQUOTA = 0
$ IF F$SEARCH("''USERDISK'[0,0]QUOTA.SYS") .EQS. "" THEN GOTO NQO
$ DQUOTA = 1
```

(Continued on next page)

Example 5–4 (Cont.) Command Procedure for Adding a User

```
$ !
$ ! Request disk quota entry information
$ !
$ INQUIRE QUOTA "Disk Quota [1000]"
$ IF QUOTA .EQS. "" THEN QUOTA = 1000
$ INQUIRE OVERDRAFT "Overdraft Quota [100]"
$ IF OVERDRAFT .EQS. "" THEN OVERDRAFT = 100
$ OPEN/WRITE FILE SYS$LOGIN:ADDQUOTA.TMP
$ WRITE FILE "RUN SYS$SYSTEM:DISKQUOTA"
$ !
$ !Add quota entry
$ !
$ WRITE FILE "USE ''USERDISK'"
$ WRITE FILE "ADD ",UIC,"/PERMQUOTA=",QUOTA,"OVERDRAFT=",OVERDRAFT
$ CLOSE FILE
$ @SYS$LOGIN:ADDQUOTA.TMP
$ DELETE SYS$LOGIN:ADDQUOTA.TMP;*/NOLOG
$NQO:
$ !
$ ! Create a first-level directory for the account
$ !
$CREATE/DIRECTORY/OWNER_UIC='UIC'/PROTECTION=(S=RWE,O=RWE,G=RE,W)-
      'USERDISK'['USERDIR']/LOG
$ IF F$SEARCH("SYS$MANAGER:LGISAMPL.COM") .EQS. "" THEN GOTO ADDACC
$ !
$ ! Copy the LOGIN.COM template to user account
$ !
$ COPY SYS$MANAGER:LGISAMPL.COM 'USERDISK'['USERDIR']LOGIN.COM
$ SET FILE/OWNER_UIC='UIC' 'USERDISK'['USERDIR']LOGIN.COM;  !Change the UIC
$ !
$ ! Add the account record to the UAF
$ !
$ADDACC:
$ OPEN/WRITE FILE SYS$LOGIN:ADDUAF.TMP
$ WRITE FILE "RUN SYS$SYSTEM:AUTHORIZE"
$ WRITE FILE "ADD ",USERNAME,"/OWNER=""",FULLNAME,"""/ACCOUNT=",ACCOUNT,-
      "/DEVICE=' 'USERDISK'/DIRECTORY=[''USERDIR']/UIC=",UIC,PRIVS,=
      "/PASSWORD=",PASSWORD
$ CLOSE FILE
$ @SYS$LOGIN:ADDUAF.TMP
$ DELETE SYS$LOGIN:ADDUAF.TMP;*/NOLOG
```

(Continued on next page)

Example 5–4 (Cont.) Command Procedure for Adding a User

```
$ !
$ ! Restore prior working environment
$ !
$CLEANUP:
$ SET TERMINAL/ECHO
$ PREVPRIV = F$SETPRV(PREVPRIV)
$ SET DEFAULT 'OLDDIR'
$ EXIT
$ !
$ ! In case proper privileges are not set
$ !
$NOPRIV:
$ WRITE SYS$OUTPUT "You need SETPRV or SYSPRV privilege to run this procedure"
$ GOTO CLEANUP
```

5.5 Maintaining the User Environment

As the work requirements of your system change, you may need to

- Create additional default records to serve as templates for new categories of users
- Delete or disable the accounts of users who leave your site
- Impose login restrictions to limit system use by certain accounts

With the Authorize Utility, you can perform these maintenance operations by modifying or deleting records in the UAF.

5.5.1 Creating Additional Default Record Templates

On systems where all users perform the same type of work, you typically use the system-supplied default record, DEFAULT, as the template for adding new user records. You may find, however, that your system supports several different user categories, each category performing a specific type of work and requiring unique record attributes. Rather than always using the system-supplied default record as a template and making numerous changes each time you add a user record, you can

create additional default UAF records to serve as templates for each user category.

Before you create additional default records, you must first decide

- What the individual user categories are
- What attributes are common to each category
- What to name the default records

Once you define a user category and establish which record attributes are needed, you can create the default record. For example, the following command creates a default record for a category of user that requires a special captive account:

```
UAF> ADD DEFAULT2/LGICMD=ALT_COM_PROC/FLAGS=CAPTIVE -
_/DEVICE=USER3:/DIRECTORY=[PRODUCT]
```

The command in this example uses the system-supplied default record DEFAULT to create the record DEFAULT2, and changes the LGICMD, login flags, default device, and default directory fields. The AUTHORIZE command COPY can then be used to create additional records having the same attributes as DEFAULT2.

The COPY command creates a new UAF record that duplicates the specified default record except where you explicitly override field values. For example, you could use the following command to create a record for a new user that duplicates the default record DEFAULT2:

```
UAF> COPY DEFAULT2 PALOOKA/PASSWORD=W7YA84MI/UIC=[360,114]
```

This example uses DEFAULT2 as a template to create a duplicate record for the user PALOOKA. Notice that the only values that are changed are those for password and UIC.

5.5.2 Deleting a User Account

The main problem in deleting an account, especially an interactive account, is cleaning up the files used by the account. The following steps are suggested:

1 Copy (or have the outgoing user of the account copy) any files of value to the ownership of another account. You should be sure to change the owner UIC of the files to match the owner UIC of the new owner. You can also use the Backup Utility (BACKUP) to copy the files to a backup tape or disk.

2 Delete the account's files and directories from the bottom up, using the following procedure:

   a Locate and examine all subdirectories using the DCL command DIRECTORY [default...], where default is the name of the account's default directory.

   b Delete the files in each subdirectory and then delete the subdirectory.

   c Delete the account's first-level directory. You cannot delete a subdirectory without first deleting the files in it. Example 5–5 illustrates a command procedure that deletes an account's files from the bottom up.

3 Delete the account, using the Authorize Utility.

4 Remove the user's disk quota entry from the disk quota file, if one existed, with the Disk Quota Utility.

If you never assign multiple users the same UIC, you can use the Backup Utility (see the *VAX/VMS Utilities Reference Volume*) to remove the user's files, even if they are scattered about the directory structure. You would use a command of this form:

```
$ BACKUP/DELETE PUBLIC:[...]/OWNER=[uic] MTAO:FRED
```

This BACKUP command copies and deletes only those files owned by the specified UIC on disk PUBLIC:. The files are copied into a save set named FRED on device MTA0.

Example 5–5 Command Procedure Template for Deleting an Account's Files

```
$ !     DELTREE.COM - deletes a complete directory tree
$ !
$ !     P1 = pathname of root of tree to delete
$ !
$ !     All files and directories in the tree, including
$ !     the named root, are deleted.
$ !
$ IF "''DELTREE'" .EQS. "" THEN DELTREE = "@SYS$LIBRARY:DELTREE"
$ ON CONTROL_Y THEN GOTO DONE
$ ON WARNING THEN GOTO DONE
$ DEFAULT = F$LOGICAL("SYS$DISK") + F$DIRECTORY()
$10:
$ IF P1 .NES. "" THEN GOTO 20
$ INQUIRE P1 "Root"
$ GOTO 10
$20:
$ IF F$PARSE(P1) .EQS. "" THEN OPEN FILE 'P1'
$ SET DEFAULT 'P1'
$LOOP:
$ FILESPEC = F$SEARCH("*.DIR;1")
$ IF FILESPEC .EQS. "" THEN GOTO LOOPEND
$ DELTREE [.'F$PARSE(FILESPEC,,,"NAME")']
$ GOTO LOOP
$LOOPEND:
$ IF F$SEARCH("*.*;*") .NES. "" THEN DELETE *.*;*
$ DIR = (F$DIRECTORY()-"]"-">")-(F$PARSE("[-]",,,-
        "(DIRECTORY")-"]"-")-"."
$ SET PROTECTION=WORLD:RWED [-]'DIR'.DIR;1
$ DELETE [-]'DIR'.DIR;1
$DONE:
$ SET DEFAULT 'DEFAULT'
```

5.5.3 Disabling a User Account

If it becomes necessary to disable an account, but deleting it at present would be undesirable, use AUTHORIZE to set the disable user flag (/FLAGS=DISUSER). If the user is logged in, the account is disabled only after the user logs out.

Disabling a powerful yet infrequently used account provides an extra security measure by eliminating the risk of guessed or stolen passwords.

5.5.4 Restricting the Use of Accounts

Often workload schedules dictate the days and times your system is used to perform specific operations. Depending on the nature of the work performed at your site, you may want to control when certain users are allowed to log in. Using the Authorize Utility, you can place controls in the login characteristics fields of the UAF record to restrict the days and times a user can log in and to inhibit certain login functions.

For a detailed description of the qualifiers used to restrict the use of accounts, see the discussion of the Authorize Utility in the *VAX/VMS Utilities Reference Volume.* The next sections discuss some of the most common restrictions.

5.5.4.1 Setting Day Types

You restrict the use of certain accounts by defining the days of the week as either PRIMARY or SECONDARY, and then assigning login restrictions to these defined day types. For example, if you define the days Saturday and Sunday as SECONDARY days, then any restrictions you assign to the SECONDARY day type apply to both.

There are two types of login restrictions that you can assign to either day type:

- Time restrictions—Limit logins to specific hours of the day
- Function restrictions—Limit types of login

By default, in every user record, the five weekdays (Monday through Friday) are defined as PRIMARY days, and the two weekend days (Saturday and Sunday) are defined as SECONDARY days.

The way you define days and assign restrictions depends on your site. For example, suppose that on weekdays your system supports a large number of interactive users, but on weekends it is used for certain operations that require dedicated system resources. By assigning restrictions to the SECONDARY day type, you can restrict users from accessing the system during the days defined as SECONDARY. You can change these day type definitions for any account using the following AUTHORIZE qualifier:

```
/PRIMEDAYS=([NO]day[,...])
```

The /PRIMEDAYS qualifier uses a list of day names to define the PRIMARY and SECONDARY days of the week. To define a day as a SECONDARY day, use the prefix NO before the day name. Any days that you omit from the list take their default value.

5.5.4.2 Restricting Logins to Specific Times

By default, there are no restrictions on login hours. You can specify login time restrictions using the following AUTHORIZE qualifiers:

Qualifier	Meaning
/[NO]ACCESS	Specifies access hours for all modes of logins
/[NO]DIALUP	Specifies access hours for interactive logins via dialup terminals
/[NO]INTERACTIVE	Specifies access hours for interactive logins via any terminal
/[NO]LOCAL	Specifies access hours for interactive logins via local terminals
/[NO]REMOTE	Specifies access hours for interactive logins via network remote terminals (SET HOST)

5.5.4.3 Restricting Login Functions

In addition to specifying hourly login restrictions, you can also assign function restrictions to an account by using appropriate keywords with the /FLAGS qualifier in the Authorize Utility. By default, there are no restrictions. Options are

Keyword	Meaning
[NO]AUDIT	[Do not] audit all security-relevant actions
[NO]CAPTIVE	[Do not] prevent user from changing any defaults at login
[NO]DEFCLI	[Do not] prevent user from changing default CLI or CLI tables
[NO]DISCTLY	[Do not] disable CTRL/Y interrupts

Keyword	Meaning
[NO]DISNEWMAIL	[Do not] suppress "New Mail ..." announcements
[NO]DISREPORT	[Do not] report login information (last login date, login failures, and so on)
[NO]DISUSER	[Do not] disable the account completely
[NO]DISWELCOME	[Do not] suppress "Welcome to ..." login message
[NO]GENPWD	[Do not] require user to use generated passwords
[NO]LOCKPWD	[Do not] prevent user from changing password
[NO]MAIL	[Do not] prevent mail delivery to the user
[NO]PWD_EXPIRED	[Do not] mark password as expired
[NO]PWD2_EXPIRED	[Do not] mark second password as expired

5.6 UAF Login Checks

To help you understand the effect of login restrictions, this section explains how the system checks the login fields of the UAF when a user attempts to log in.

When a user activates a terminal (by turning it on and pressing RETURN if directly connected, or by dialing in to a system and observing the remote connect protocol), and that terminal is not allocated by a user process, the system prompts for a name and password. The person using the terminal must type a name and password combination that exists in a UAF record, or the system denies him further access. If the name and password are accepted, the system then performs the following operations:

1. Examines the login flags, beginning with DISUSER. If DISUSER is set, the login attempt fails. Note that setting this flag for powerful, infrequently used accounts (such as SYSTEM, SYSTEST, and FIELD) virtually eliminates the risk of guessed passwords for those accounts.

2. If the DISUSER flag is not set, verifies primary or secondary day restrictions. After checking the current day type, the system determines whether hourly login restrictions are in effect (as defined by the /ACCESS, /DIALUP,

/INTERACTIVE, /LOCAL, and /REMOTE qualifiers). If the current hour is restricted, the login fails immediately. Otherwise it succeeds.

3 If the login is successful, passes control to the command interpreter (for example, DCL) named in the user's UAF record.

4 Checks whether SYS$SYLOGIN is defined. If so, the logical name is translated (in most cases to SYS$MANAGER:SYLOGIN.COM) and that procedure is executed. When the procedure completes, the system searches for the name of a login command procedure in that user's UAF record. If a command procedure is specified in the LGICMD field and that procedure exists, it is executed. Otherwise, if the LGICMD field is blank, the user's command file named LOGIN is executed automatically (if it exists).

After a successful login, the command interpreter prompts for user input (DCL usually displays a dollar sign) and the user responds with commands acceptable to the command interpreter. (DCL accepts those commands documented in the *VAX/VMS DCL Dictionary.*) However, the system prohibits activities that violate the user's privilege allowance or exceed resource quotas.

5.7 Alternate Login Procedures

There are situations when normal login activities using the UAF are either impossible or undesirable. The UAF may be inaccessible for some reason, or you may want to boot for stand-alone functions using an alternate UAF that restricts access to the system. This section discusses these two situations.

If the UAF is locked, disabled, or not present, you can log in on the console terminal with any name and password. The system assigns the following values to your user account:

- Name—Your user name
- UIC—[001,004]
- Command interpreter—DCL
- Login flags—None

- Priority—Value of the system parameter DEFPRI
- Resources—Values of the PQL system parameters
- Privileges—All

The process name is normally the name of the device on which you logged in, that is, _OPA0:.

If you want to take the system for stand-alone use, you can select, at bootstrap time, an alternate UAF named SYS$SYSTEM:SYSUAFALT.DAT by using a system parameter file in which the UAFALTERNATE parameter has a value of 1. You can also change this value to 1 during a conversational bootstrap operation. (See Chapter 11 for information on creating system parameter files.) Note that a file named SYSUAFALT in UAF format must exist at bootstrap time. (The system initially changes to the alternate UAF by assigning SYSUAF as a logical name for SYSUAFALT.) If the UAFALTERNATE parameter is set at bootstrap time and no alternate UAF exists, the system behaves like an *open system*.

You can create or modify an alternate UAF with the following commands:

```
$ SET DEFAULT SYS$SYSTEM
$ DEFINE SYSUAF SYSUAFALT
$ RUN AUTHORIZE
```

Any time after booting the system with an alternate UAF, you can switch to the standard UAF (that is, SYS$SYSTEM:SYSUAF.DAT) with the following DCL command:

```
$ DEASSIGN SYSUAF/SYSTEM
```

You can also switch to the alternate UAF after bootstrap time with the following DCL command:

```
$ DEFINE/SYSTEM/EXEC SYSUAF SYSUAFALT
```

(For this mode of operation, the alternate UAF can have any filename.)

5.8 Creating Proxy Accounts

A *proxy login* account allows users on a remote node in a network to access data by way of a local account on your system. Proxy accounts are useful when you want to grant one or more users on a remote node access to specific files but find it undesirable to provide them with a private account on your system. You establish and control proxy accounts with the Authorize Utility.

With proxy accounts, you can authorize one or more users on a remote node to issue DCL commands that access data from a particular account on your system. Proxy accounts allow remote users to access specific local data (for example, type and print files) without having to log in to your system or use an access control string. Remote users assume the same file access rights as the local account and also receive the default privileges of the local account. The following sections explain the procedures for setting up proxy accounts.

5.8.1 Creating a Network User Authorization File

Proxy accounts exist as records in a network UAF called SYS$SYSTEM:NETUAF.DAT and are created and maintained using the Authorize Utility. Since the network UAF is not provided with the distribution kit, you must create and initialize it. You use the AUTHORIZE command CREATE/PROXY for this operation.

Proxy accounts must be associated with a user account in the SYSUAF.DAT file on your local system. You will probably want to create a "standard access" account in the UAF for proxy accounts. For example, you could create an account named REMOTE_MARKET with limited privileges, which allows access to certain data files on your local system.

Suppose you have a group of users on your local system who prepare marketing reports, and who rely on input from users on other systems. You would assign the REMOTE_MARKET account in SYUAF.DAT the same group number and default privileges you assign to the local marketing group. In this way, the remote contributors could access any data files that are "owned" by users in your marketing group and that are not protected from GROUP access.

5.8.2 Adding Proxy Accounts

You create a proxy account by adding records to the network UAF that equate one or more users on a remote node to users on your home node.

For example, the following command adds a user proxy account:

```
UAF> ADD/PROXY HAL::WALTER REMOTE_MARKET
```

Assume that you have created an account named REMOTE_MARKET on your home node. The record created in this example permits the user WALTER on remote node HAL to access data by way of the REMOTE_MARKET account on your home node. With the proxy account, WALTER can access any data from his node that REMOTE_MARKET can access locally.

Note: Since the remote user receives the same privileges as the local user, you should not set up proxy accounts associated with local accounts that have special privilege. Granting remote users such access powers poses a threat to the security of your system.

When creating such an account, remember that each user on a remote node is allowed access to only one account on the local node. So, in the previous example, HAL::WALTER is allowed access only to the files that REMOTE_MARKET can access. If you tried to add another proxy account for HAL::WALTER, you would receive an error message.

A number of your users may have accounts on a remote node and require ready access to their local files. You can create a network authorization record that grants access to each of them, provided that the user name on your system is the same as the user name on the remote node. The following form of the ADD/PROXY command adds such a record:

```
UAF> ADD/PROXY HAL::* *
```

This command authorizes any user on the remote node HAL to access any account with the same user name on your system.

Similarly, you might want to permit this sort of access for just one user.

```
UAF> ADD/PROXY HAL::BARBARA *
```

In this case, the user BARBARA on node HAL can access any files that BARBARA (and only BARBARA) can access on your system.

5.8.3 Controlling Proxy Logins

Whenever a proxy login request occurs, the system verifies that proxy access on the participating nodes is enabled. (By default, both incoming and outgoing proxy access is enabled for each node.) If access is enabled, the system checks account information in your NETUAF.DAT. You can, however, change proxy access or disable it totally by using the Network Control Program (NCP). Use of NCP to control proxy access is described in the *Guide to Networking on VAX/VMS*.

6 Resource Control

This chapter contains detailed descriptions of the resource control attributes that you can assign to a user process when creating a record in the UAF:

- Limits on reusable system resources
- Base priority for scheduling user processes
- Privileges allowing use of restricted and sensitive system functions

In addition, this chapter discusses the accounting log file as a tool for tracking resource use.

6.1 Setting Limits on Reusable System Resources

Each user of the system is limited in the consumption of such resources as system memory. You set limits when you define the user to the system through the creation of an account in the UAF.

Limits control the way in which a process shares its allotment of a resource with the subprocesses that it creates. In addition to restricting the number of processes that a single user or account may have at any given time, the system uses four types of limits for sharing resources:

- **Pooled**—If the limit on the use of a resource is pooled, a process and created subprocesses share the total limit on a first-come, first-served basis.
- **Deductible**—If the limit on the use of a resource is deductible, a subprocess is allotted a portion of the total limit; the portion given to the subprocess is deducted from the total limit.
- **Nondeductible**—If the limit is nondeductible, the subprocess is allotted the total limit of the creating process; there is no deduction from the allotment of the creating process.

- **System-wide**—If the limit is system-wide, a process and all created subprocesses with the same username or account share the total limit on a first-come, first-served basis.

In creating a UAF record, you assign values to the limits shown in Table 6–1. These limits are described in the following sections. Usually, you simply assign the default values for these limits. However, see the *Guide to VAX/VMS Performance Management* for a discussion of how to evaluate and adjust the limits in the context of performance optimization strategies.

Table 6–1 summarizes each of these limits, the suggested value, and the type of limit.

Table 6–1 Process Resource Limits, Suggested Values, Types, and Descriptions

Limit	Value	Type[1]	Description
ASTLM	24	N	AST queue limit
BIOLM	18	N	Buffered I/O count limit
BYTLM	8192	P	I/O byte count limit
CPU	0	D	CPU time limit (0 = no limit)
DIOLM	18	N	Direct I/O count limit
ENQLM	30	P	Enqueue quota
FILLM	20	P	Open file limit
JTQUOTA	1024	P	Initial byte quota for job-wide logical name table
MAXACCTJOBS	0	S	Maximum active processes for a single account (0 = no limit)
MAXDETACH	0	S	Maximum detached processes for a single username (0 = no limit)
MAXJOBS	0	S	Maximum active processes for a single username (0 = no limit)
PGFLQUO	12800	P	Paging file limit
PRCLM	2	P	Subprocess creation limit

[1]D=deductible, N=nondeductible, P=pooled, S=system-wide

Table 6–1 (Cont.) Process Resource Limits, Suggested Values, Types, and Descriptions

Limit	Value	Type[1]	Description
SHRFILLM	0	P	Maximum number open shared files (0 = no limit)
TQELM	10	P	Timer queue entry limit
WSDEF	300	N	Default working set size
WSEXTENT	700	N	Working set extent
WSQUO	350	N	Working set quota

[1]D=deductible, N=nondeductible, P=pooled, S=system-wide

6.1.1 AST Queue Limit (ASTLM)

The AST queue limit (ASTLM) limits the sum of the following:

- The number of asynchronous system trap (AST) requests that a user's process can have outstanding at one time
- The number of scheduled wakeup requests that a user's process can have outstanding at one time

This limit affects not only all system services that accept an AST address as an argument, but also the Schedule Wakeup ($SCHDWK) system service.

If the deferred write option (DFW) is enabled, the number of ASTs used per file is equal to 1, plus the number of record streams, plus the multibuffer count. Otherwise, the number is 1 plus the number of record streams.

ASTLM is a nondeductible limit with a suggested typical value of 24.

6.1.2 Buffered I/O Count Limit (BIOLM)

The buffered I/O count limit (BIOLM) limits the number of outstanding buffered I/O operations permitted a user's process.

In a buffered I/O operation, the data transfer takes place from an intermediate buffer in the system pool, not from a process-specified buffer. Buffered operations for a user process include terminal I/O, file system and network I/O, card reader input, and unspooled printer output. During a buffered I/O operation, the pages containing the process-specified buffer need not be locked in memory.

BIOLM is a nondeductible limit with a suggested typical value of 18.

6.1.3 Buffered I/O Byte Count Limit (BYTLM)

The buffered I/O byte count limit (BYTLM) limits the amount of buffer space that a user's process can use.

This buffer space is used for buffered I/O operations and for the creation of temporary mailboxes. It also limits the number of mapping windows the user can create as segmented (or cathedral) windows. Cathedral windows are primarily useful to reduce the overhead required to read large files.

BYTLM is a pooled limit with a suggested typical value of 8192.

6.1.4 CPU Time Limit (CPU)

The CPU time limit (CPU) limits the amount of CPU time that a user's process can use per interactive session or batch job.

The time must be specified in abbreviated delta format—hh:mm:ss:cc.

CPU is a deductible limit with a suggested typical value of 0 (no limit), but the value only applies to this instance or other instances of the user's processes. CPU is not cumulative across separate sessions or batch jobs.

6.1.5 Direct I/O Count Limit (DIOLM)

The direct I/O count limit (DIOLM) limits the number of outstanding direct I/O operations permitted a user's process.

In a direct I/O operation, the data transfer takes place directly from a process-specified buffer. Direct I/O operations for a user process typically include disk and tape I/O. The pages containing this buffer are locked in memory by the operating system during the direct I/O operation.

DIOLM is a nondeductible limit with a suggested typical value of 18.

6.1.6 Enqueue Quota (ENQLM)

The enqueue quota (ENQLM) limits the number of locks a process (and its subprocesses) can own. VAX Record Management Services (RMS) uses the Lock Management Facility to synchronize shared file access, global buffers, and record locks. Because VAX RMS takes out one lock for every shared file, local buffer, global buffer section, and outstanding record lock, users who expect to perform large amounts of VAX RMS file sharing should have ENQLM set to a large value.

If your process performs extensive VAX RMS file sharing without sufficient enqueue quota, you could receive the SS$_EXENQLM error message. Furthermore, if your system performs extensive VAX RMS file sharing and the value of the LOCKIDTBL system parameter is too low, you could receive the SS$_NOLOCKID error message. Note that whenever you increase the value of LOCKIDTBL, you may have to increase the value of the RESHASHTBL system parameter (see the discussion of the System Generation Utility (SYSGEN) in the *VAX/VMS Utilities Reference Volume*).

For shared files, the value of ENQLM should represent the number of files open as shared multiplied by the number of locks per process per file. If you use the default multibuffer counts, you can estimate the number of locks as 4 for indexed sequential files and 3 for relative files. If you use other than the default value for the multibuffer counts, you can estimate the number of locks per process per file as one per file plus the multibuffer count for that file plus the number of records

locked (which is usually one). Use the DCL command SHOW RMS_DEFAULT to display the default multibuffer counts.

ENQLM is a pooled limit with a suggested typical value of 30.

6.1.7 Open File Limit (FILLM)

The open file limit (FILLM) limits the number of files that a user's process can have open at one time. This limit includes the number of network logical links that can be active at the same time.

FILLM is a pooled limit with a suggested typical value of 20. Note that each open file also requires at least 96 bytes of BYTLM.

6.1.8 Job Table Quota (JTQUOTA)

The job table quota (JTQUOTA) specifies the initial byte quota with which the job-wide logical name table is to be created.

JTQUOTA is a pooled quota with a suggested typical value of 1024.

6.1.9 Maximum Account Jobs Limit (MAXACCTJOBS)

The maximum account jobs limit (MAXACCTJOBS) specifies the maximum number of batch, interactive, and detached processes that may be active at one time for all users of a single account.

MAXACCTJOBS is a system-wide limit with a suggested typical value of 0.

6.1.10 Maximum Detached Processes Limit (MAXDETACH)

The maximum detached processes limit (MAXDETACH) specifies the maximum number of detached processes that may be active at one time for a single username. Processes that exceed this limit are terminated.

MAXDETACH is a system-wide limit with a suggested typical value of 0.

6.1.11 Maximum Process Jobs Limit (MAXJOBS)

The maximum process jobs limit (MAXJOBS) specifies the maximum number of interactive, batch, and detached processes that can be active at one time for a username. Processes that exceed this limit are terminated.

MAXJOBS is a system-wide limit with a suggested typical value of 0.

6.1.12 Paging File Limit (PGFLQUO)

The paging file limit (PGFLQUO) limits the number of pages that the user's process can use in the system paging file. The paging file provides temporary disk storage for pages forced out of memory by a memory management operation. PGFLQUO limits the total virtual address space that can be created using the Create Virtual Address Space ($CRETVA) or Expand Program/Control Region ($EXPREG) system services.

PGFLQUO is a pooled limit with a suggested typical value of 12800.

6.1.13 Subprocess Creation Limit (PRCLM)

The subprocess creation limit (PRCLM) limits the number of subprocesses a user's process can create.

The process that is created when a user logs in to the system can in turn create subprocesses. These subprocesses are all accountable to the user and share the resources allotted to the initial process.

PRCLM is a pooled limit with a suggested typical value of 2.

6.1.14 Shared Files Limit (SHRFILLM)

The shared files limit (SHRFILLM) specifies the maximum number of shared files that an account may have open at one time. The shared files limit is a pooled limit with a suggested typical value of 0.

6.1.15 Timer Queue Entry Limit (TQELM)

The timer queue entry limit (TQELM) limits the sum of the following:

- The number of entries that a user's process can have in the timer queue
- The number of temporary common event flag clusters that a user's process can have

This limit does not govern the creation of permanent event flag clusters.

Timer queue entries are used in time-dependent scheduling; common event flags are used in synchronizing activities among groups of cooperating processes.

TQELM is a pooled limit with a suggested typical value of 10.

6.1.16 Default Working Set Size (WSDEF)

The default working set size (WSDEF) sets the initial working set size limit for a user's process.

WSDEFAULT is a nondeductible limit with a suggested typical value of 300 pages. If the value specified exceeds the value of WSQUOTA (see Section 6.1.18), the lesser value is used.

6.1.17 Working Set Extent (WSEXTENT)

The working set extent (WSEXTENT) specifies the maximum size to which a user's physical memory usage can grow, independent of the system load. This enlargement of the physical memory for a user is accomplished by the Adjust Working Set Limit ($ADJWSL) system service, and is normally done for the user by VAX/VMS in response to heavy page faulting by the user.

WSEXTENT is a nondeductible quota with a suggested typical value of 700 or more. This value should always be greater than or equal to WSQUOTA. The value is controlled by the system parameter WSMAX.

6.1.18 Working Set Quota (WSQUO)

The user's physical memory usage can grow on a typically loaded system. That is, this parameter guarantees the user that the number of physical pages specified will be available. For example, WSQUOTA limits the number of pages a user can lock in memory.

WSQUOTA is a nondeductible quota with a suggested typical value of 350. This value should be greater than or equal to WSDEFAULT. The value is controlled by the system parameter WSMAX.

6.2 Setting Priorities for User Processes

A user's priority is the base priority used in scheduling the process that the system creates for the user. There are 32 levels of software priority in the VAX/VMS system, 0 through 31. The highest priority is 31; the lowest is 0. The range of priorities for timesharing processes is 1 through 15; the range for real-time processes is 16 through 31.

Processes with real-time priorities are scheduled strictly according to base priority; in other words, the executable real-time process with the highest base priority is executed first. Processes with timesharing priorities are scheduled according to a slightly different principle, to promote overlapping of computation and I/O activities.

In the user's account record of the UAF, the default value of a user's priority is 4; for practical purposes, the minimum value is 1. You should ensure that the priority for timesharing users remains at the default. Note that if you give some users an advantage over other users by raising their priorities, ragged performance will result, because the system reacts sharply to even small priority differences.

Never specify a value over 31 (system operation will be unpredictable).

6.3 Assigning Privileges

Privileges restrict the performance of certain system activities to certain users. These restrictions protect the integrity of the operating system's performance and thus the integrity of service provided to users. You should grant privileges to each user on the basis of two factors:

- Whether the user has the skill and experience to use the privilege without disrupting the system
- Whether the user has a legitimate need for the privilege

Privileges fall into seven categories according to the damage that the user possessing them could cause the system:

- None—No privileges
- Normal—Minimum privileges to use the system effectively

- Group—Potential to interfere with members of the same group
- Devour—Potential to consume noncritical system-wide resources
- System—Potential to interfere with normal system operation
- Files—Potential to compromise file security
- All—Potential to control the system

A user cannot execute an image that requires a privilege he or she does not possess, unless the image is installed as a known image with the privilege in question. (See Chapter 8 for instructions on installing known images.) Execution of a known image with tempory privileges (for the duration of the image's execution) grants those privileges to the user process executing the image. Thus, you should install user images with amplified privileges only after ensuring that the user needs the access and is unlikely to misuse it.

A user's privileges are recorded in the user's UAF record in a 64-bit privilege vector. When a user logs in to the system, the user's privilege vector is stored in the header of the user's process. In this way, the user's privileges are passed on to the process created for the user. Users can use the DCL command SET PROCESS/PRIVILEGES to enable and disable privileges for which they are authorized, to further control the privileges available to the images they run. Moreover, any user with the SETPRV privilege can enable any privilege.

Table 6–2 lists the privileges by category and gives brief, general definitions of them. The following sections describe each privilege in detail in alphabetical order and indicate privilege categories.

Table 6–2 VAX/VMS Privileges by Category, with Definitions

Category	Privilege	Activity Permitted
None	None	None requiring privileges
Normal	MOUNT	Execute mount volume QIO
	NETMBX	Create network connections
	TMPMBX	Create temporary mailbox
Group	GROUP	Control processes in the same group
	GRPPRV	Group access via SYSTEM protection field
Devour	ACNT	Disable accounting
	ALLSPOOL	Allocate spooled devices
	BUGCHK	Make bugcheck error log entries
	EXQUOTA	Exceed disk quotas
	GRPNAM	Insert group logical names in the name table
	PRMCEB	Create/delete permanent common event flag clusters
	PRMGBL	Create permanent global sections
	PRMMBX	Create permanent mailboxes
	SHMEM	Create/delete structures in shared memory
System	ALTPRI	Set base priority higher than allotment
	OPER	Perform operator functions
	PSWAPM	Change process swap mode
	SHARE	Access devices allocated to other users
	SYSLCK	Lock system-wide resources
	WORLD	Control any process
Files	DIAGNOSE	Diagnose devices
	SYSGBL	Create system-wide global sections
	VOLPRO	Override volume protection

Table 6–2 (Cont.) VAX/VMS Privileges by Category, with Definitions

Category	Privilege	Activity Permitted
All	BYPASS	Disregard protection
	CMEXEC	Change to executive mode
	CMKRNL	Change to kernel mode
	DETACH	Create detached processes of arbitrary UIC
	LOG_IO	Issue logical I/O requests
	PFNMAP	Map to specific physical pages
	PHY_IO	Issue physical I/O requests
	READALL	Possess read access to everything
	SECURITY	Perform security-related functions
	SETPRV	Enable any privilege
	SYSNAM	Insert/delete system logical names in the name table
	SYSPRV	Access objects via SYSTEM protection field

6.3.1 ACNT Privilege (Devour)

The ACNT privilege allows a user to create subprocesses or detached processes in which accounting is disabled. Thus, only such a privileged user can issue the DCL command RUN with the /NOACCOUNTING qualifier or inhibit accounting in the Create Process ($CREPRC) system service.

6.3.2 ALLSPOOL Privilege (Devour)

The ALLSPOOL privilege allows the user's process to allocate a spooled device by executing the Allocate Device ($ALLOC) system service. This service lets a process allocate, or reserve, a device for its exclusive use. A shareable mounted device cannot be allocated. The user may also allocate a spooled device by using the DCL command ALLOCATE.

You should grant this privilege only to users who need to perform logical or physical I/O operations to a spooled device. Ordinarily, the privilege of allocating a spooled device is granted only to symbionts (see Section 9.3.8).

6.3.3 ALTPRI Privilege (System)

The ALTPRI privilege allows the user's process to

- Increase its own base priority
- Set the base priority of another process to a value higher than that of the target process

The base priority is increased by executing the Set Priority ($SETPRI) system service or the DCL command SET PROCESS /PRIORITY. As a rule, this system service lets a process set its own base priority or the base priority of another process. However, one process can set the priority of a second process if

- The process calling the $SETPRI system service has the same UIC as the target process
- The calling process has process control privilege (GROUP or WORLD) over the target process

With ALTPRI, a process can create a process with a priority higher than its own. It creates such a process by using an optional argument to the Create Process ($CREPRC) system service or to the DCL command RUN.

You should not grant this privilege widely; if unqualified users have the unrestricted ability to set base priorities, the fair and orderly scheduling of processes for execution can easily be disrupted.

6.3.4 BUGCHK Privilege (Devour)

Use of the BUGCHK privilege should be restricted to system software supplied by DIGITAL that uses the VAX/VMS Bugcheck Facility. This privilege allows the user process to make bugcheck error log entries.

6.3.5 BYPASS Privilege (All)

The BYPASS privilege allows the user's process read, write, execute, and delete access to all files, bypassing UIC protection.

You should grant this privilege with extreme caution, as it overrides all file protection. It should be reserved for experienced users, well-tested, reliable programs and command procedures, or the system backup operation (see Chapter 2 and the *Guide to VAX/VMS Disk and Magnetic Tape Operations* for a discussion of backup operations). SYSPRV is adequate for interactive use, as it ultimately grants access to all files, while still providing access checks.

6.3.6 CMEXEC Privilege (All)

The CMEXEC privilege allows the user's process to execute the Change Mode to Executive ($CMEXEC) system service.

This system service lets a process change its access mode to executive, execute a specified routine, and then return to the access mode that was in effect before the system service was called. While in executive mode, the process is allowed to execute the Change Mode to Kernel ($CMKRNL) system service.

You should grant this privilege only to users who need to gain access to protected and sensitive data structures and internal functions of the operating system. If unqualified users have unrestricted access to sensitive data structures and functions, the operating system and service to other users can easily be disrupted. Such disruptions can include failure of the system, destruction of the database, and exposure of confidential information to unauthorized persons.

6.3.7 CMKRNL Privilege (All)

The CMKRNL privilege allows the user's process to execute the Change Mode to Kernel ($CMKRNL) system service.

This system service lets a process change its access mode to kernel, execute a specified routine, and then return to the access mode that was in effect before the system service was called.

In granting this privilege, follow the guidelines for the CMEXEC privilege.

6.3.8 DETACH Privilege (All)

The DETACH privilege allows the user's process to create detached processes by executing the Create Process ($CREPRC) system service. Detached processes remain in existence even after the user who created them has logged off the system.

An example of a detached process is the process created by the system for a user when the user logs in to the system.

There is no restriction on the UIC that can be specified for a detached process. Thus, there are no restrictions on the files and directories to which a detached process can gain access.

6.3.9 DIAGNOSE Privilege (Files)

The DIAGNOSE privilege allows the user to run online diagnostic programs and to intercept and copy all messages that are written to the error log file.

6.3.10 EXQUOTA Privilege (Devour)

The EXQUOTA privilege allows the space taken by the user's files on given disk volumes to exceed any usage quotas set for the user (as determined by UIC) on those volumes.

6.3.11 GROUP Privilege (Group)

The GROUP privilege allows the user's process to affect other processes in its own group by executing the following process control system services:

- Cancel Wakeup ($CANWAK)
- Delete Process ($DELPRC)
- Force Exit ($FORCEX)
- Resume Process ($RESUME)
- Schedule Wakeup ($SCHDWK)
- Set Priority ($SETPRI)
- Suspend Process ($SUSPND)
- Wake ($WAKE)

The user's process is also allowed to examine other processes in its own group by executing the Get Job/Process Information ($GETJPI) system service. A user with the GROUP privilege can issue the following DCL commands for other processes in its group: SET QUEUE, DELETE/ENTRY, STOP/ENTRY, and SET PROCESS.

The GROUP privilege is not needed for a process to exercise control over, or to examine, subprocesses that it created. You should grant this privilege to users who need to exercise control over each other's processes and operations.

6.3.12 GRPNAM Privilege (Devour)

The GRPNAM privilege allows the user's process to insert names in the logical name table of the group to which the process belongs and to delete names from that table, using the following logical name system services: Create Logical Name ($CRELNM) and Delete Logical Name ($DELLNM).

In addition, the privileged user can use the DCL commands ASSIGN and DEFINE to add names to the group logical name table. The DEASSIGN command deletes names from the table, and the /GROUP qualifier of the MOUNT command shares volumes among group members.

This privilege should not be granted to all users of the system because it allows a user to create an unlimited number of group logical names. When unqualified users have the unrestricted ability to create group logical names, excessive use of system dynamic memory can degrade system performance. In addition, a user with the GRPNAM privilege can interfere with the activities of other users in the same group by creating definitions of commonly used logical names, such as SYS$SYSTEM.

6.3.13 GRPPRV Privilege (Group)

The GRPPRV privilege allows a process access to files using the files' SYSTEM protection field when the process's group matches the group of the file owner. It also allows a process to change the protection of files whose owner group matches the process's group.

6.3.14 LOG_IO Privilege (All)

The LOG_IO privilege allows the user's process to execute the Queue I/O Request ($QIO) system service to perform logical-level I/O operations. This privilege is also required for certain device control functions, such as setting permanent terminal characteristics.

Usually, user I/O requests are handled indirectly by use of an I/O package, such as the VAX Record Management Services. However, to increase their control over I/O operations and to improve the efficiency of I/O operations, skilled users sometimes prefer to handle directly the interface between their process and a system I/O driver program. They can do this by executing the Queue I/O Request system service; in many instances, the operation called for is a logical-level I/O operation.

You should grant this privilege only to users who need it, because it allows a process to access data anywhere on the selected volume without the benefit of any file structuring. If this privilege is given to unqualified users who have no need for it, the operating system and service to other users can easily be disrupted. Such disruptions can include the destruction of information on the system device, the destruction of user data,

and the exposure of confidential information to unauthorized persons.

6.3.15 MOUNT Privilege (Normal)

The MOUNT privilege allows the user's process to execute the mount volume QIO function. The use of this function should be restricted to system software supplied by DIGITAL.

6.3.16 NETMBX Privilege (Normal)

The NETMBX privilege allows the user to perform functions related to a DECnet computer network. This privilege is normally granted to general users.

6.3.17 OPER Privilege (System)

The OPER privilege allows use of the operator communication process (OPCOM), as follows:

- To reply to user requests
- To broadcast messages to all terminals logged in
- To designate terminals as operators' terminals and specify the types of messages to be displayed
- To initialize and control the operator log file

In addition, this privilege lets the user set devices spooled, create and control both batch and output queues, and initialize and mount public volumes.

You should grant this privilege only to operators of the system. These are the users who respond to the requests of ordinary users, who tend to the needs of the system's peripheral devices (mounting reels of tape and changing printer forms), and who attend to all the other day-to-day chores of system operation. (A nonprivileged user can log in on the console terminal to respond to operator requests, for example, to mount a tape.)

6.3.18 PFNMAP Privilege (All)

The PFNMAP privilege allows the user's process to map to specific pages of physical memory or I/O device registers, no matter who is using the pages or registers.

You should exercise caution in granting this privilege. If unqualified users have unrestricted access to physical memory, the operating system and service to other users can easily be disrupted. Such disruptions can include failure of the system, destruction of the database, and exposure of confidential information to unauthorized persons.

6.3.19 PHY_IO Privilege (All)

The PHY_IO privilege allows the user's process to execute the Queue I/O Request ($QIO) system service to perform physical-level I/O operations.

Usually, users' I/O requests are handled indirectly by use of an I/O package such as the VAX Record Management Services. However, to increase their control over I/O operations and to improve the efficiency of their applications, skilled users sometimes prefer to handle directly the interface between their process and a system I/O driver program. They can do this by executing the Queue I/O Request system service; in many instances, the operation called for is a physical-level I/O operation.

You should grant the PHY_IO privilege only to users who need it; in fact, this privilege should be granted even more carefully than the LOG_IO privilege (see Section 6.3.14). If this privilege is given to unqualified users who have no need for it, the operating system and service to other users can easily be disrupted. Such disruptions can include the destruction of information on the system device, the destruction of user data, and the exposure of confidential information to unauthorized persons.

6.3.20 PRMCEB Privilege (Devour)

The PRMCEB privilege allows the user's process to create or delete a permanent common event flag cluster by executing the Associate Common Event Flag Cluster ($ASCEFC) or Delete Common Event Flag Cluster ($DLCEFC) system service. Common event flag clusters enable cooperating processes to communicate with each other, thus providing the means of synchronizing their execution.

This privilege should not be granted to all users of the system, because it allows the user to create an unlimited number of permanent common event flag clusters. A permanent cluster remains in the system even after the creating process has been terminated and continues to use up a portion of system dynamic memory. When many users have the unrestricted ability to create permanent common event flag clusters, the excessive use of system dynamic memory can degrade system performance.

6.3.21 PRMGBL Privilege (Devour)

The PRMGBL privilege allows the user's process to create global sections by executing the Create and Map Section ($CRMPSC) system service. In addition, the user with this privilege (plus the CMKRNL and SYSGBL privileges) can use the Install Utility.

Global sections are shared structures that can be mapped simultaneously in the virtual address space of many processes. All processes see the same code or data. Global sections are used for reentrant subroutines or data buffers.

You should grant this privilege with care. If permanent global sections are not explicitly deleted, they tie up space in the global section and global page tables, which are limited resources.

6.3.22 PRMMBX Privilege (Devour)

The PRMMBX privilege allows the user's process to create or delete a permanent mailbox by executing the Create Mailbox and Assign Channel ($CREMBX) system service or the Delete Mailbox ($DELMBX) system service.

Mailboxes are buffers in virtual memory that are treated as if they were record-oriented I/O devices. A mailbox is used for general interprocess communication.

The PRMMBX privilege should not be granted to all users of the system. Permanent mailboxes are not automatically deleted when the creating processes are deleted and, thus, continue to use up a portion of system dynamic memory.

6.3.23 PSWAPM Privilege (System)

The PSWAPM privilege allows the user's process to control whether it can be swapped out of the balance set by executing the Set Process Swap Mode ($SETSWM) system service. Not only must a process have this privilege to lock itself in the balance set (that is, to disable swapping), but also to unlock itself (that is, to enable swapping).

With this privilege, a process can create a process that is locked in the balance set (process swap mode disabled) by using an optional argument to the Create Process ($CREPRC) system service or, when the DCL command RUN is used to create a process, by using a qualifier of the RUN command.

You should grant this privilege only to users who need to lock a process in memory for performance reasons. Typically, this will be a real-time process. If unqualified users have the unrestricted ability to lock processes in the balance set, physical memory can be held unnecessarily, thereby degrading system performance.

6.3.24 READALL Privilege (All)

The READALL privilege allows the process to bypass existing restrictions that would otherwise prevent the process from reading a file. However, unlike the BYPASS privilege, which permits writing and deleting, READALL permits only reading of the file.

You should grant this privilege only to individuals with appropriate management responsibilities at your site.

6.3.25 SECURITY Privilege (All)

The SECURITY privilege allows a process to perform security-related functions such as enabling or disabling security audits or setting the system password.

You should grant this privilege only to individuals responsible for system security.

6.3.26 SETPRV Privilege (All)

The SETPRV privilege allows the user's process to create processes whose privileges are greater than its own, by executing the Create Process ($CREPRC) system service with an optional argument, or by issuing the DCL command RUN to create a process. A user with this privilege can also execute the DCL command SET PROCESS/PRIVILEGES to obtain any desired privilege.

You should exercise the same caution in granting the SETPRV privilege as in granting any other privilege in the ALL category, since SETPRV allows the user to enable any or all privileges.

6.3.27 SHARE Privilege (System)

The SHARE privilege allows the user's process to assign channels to devices allocated to other users.

Normally you should grant this privilege only to system software such as symbionts. This privilege would allow an irresponsible user to interfere with the operation of devices belonging to other users.

6.3.28 SHMEM Privilege (Devour)

The SHMEM privilege allows the user's process to create and delete global sections and mailboxes (permanent and temporary) in multiport memory, if the process also has appropriate PRMGBL, PRMMBX, SYSGBL, and TMPMBX privileges. Just as in local memory, the space required for a multiport memory temporary mailbox counts against the buffered I/O byte count limit (BYTLM) of the process.

6.3.29 SYSGBL Privilege (Files)

The SYSGBL privilege allows the user's process to create system global sections by executing the Create and Map Section ($CRMPSC) system service. In addition, the user with this privilege (plus the CMKRNL and PRMGBL privileges) can use the Install Utility.

You should exercise caution in granting this privilege. System global sections require space in the global section and global page tables, which are limited resources.

6.3.30 SYSLCK Privilege (System)

The SYSLCK privilege allows the user's process to lock system-wide resources with the Enqueue Lock Request ($ENQ) system service. You should grant this privilege to users who need to run programs that lock resources in the system-wide resource name space. You should exercise caution in granting this privilege, since users who hold it can interfere with the synchronization of system and other users' software.

6.3.31 SYSNAM Privilege (All)

The SYSNAM privilege allows the user's process to insert names in the system logical name table and to delete names from that table by using the Create Logical Name ($CRELNM) and Delete Logical Name ($DELLNM) system services.

In addition, the user with this privilege can use the DCL commands ASSIGN and DEFINE to add names to the system logical name table, and can use the DEASSIGN command to delete names from the table.

You should grant this privilege only to the the system operators or to system programmers who need to define system logical names (such as names for user devices, library directories, and the system directory). For example, to mount or dismount a system volume, which entails defining a system logical name, you must have the SYSNAM privilege. Note that a user with the SYSNAM privilege could redefine such critical system logical names as SYS$SYSTEM and SYSUAF, thus gaining control of the system.

6.3.32 SYSPRV Privilege (All)

The SYSPRV privilege allows the user to assume the file access rights of a system user, and to change the owner UIC and protection of a file. Even if a file is protected against SYSTEM access, the user with the SYSPRV privilege can simply change the file's protection to gain access to it.

You should exercise caution in granting this privilege. If unqualified users have SYSTEM access rights, the operating system and service to others can easily be disrupted. Such disruptions can include failure of the system, destruction of the database, and exposure of confidential information to unauthorized persons.

6.3.33 TMPMBX Privilege (Normal)

The TMPMBX privilege allows the user's process to create a temporary mailbox by executing the Create Mailbox and Assign Channel ($CREMBX) system service.

Mailboxes are buffers in virtual memory that are treated as if they were record-oriented I/O devices. A mailbox is used for general interprocess communication. Unlike a permanent mailbox, which must be explicitly deleted, a temporary mailbox is deleted automatically when it is no longer referenced by any process. Note that this privilege is required to use the DCL commands SUBMIT and PRINT.

You should usually grant this privilege to all users of the system to facilitate interprocess communication. System performance is not likely to be degraded by permitting the creation of temporary mailboxes, because their number is controlled by limits on the use of system dynamic memory (BYTLM quota).

6.3.34 VOLPRO Privilege (Files)

The VOLPRO privilege allows the user to

- Initialize a previously used volume with an owner UIC different from the user's own UIC
- Override the expiration date on a disk or disk volume he or she does not own
- Mount with the /FOREIGN qualifier a Files–11 volume he or she does not own
- Override the owner UIC protection of a volume

The VOLPRO privilege permits control only over volumes that the user can mount or initialize. Volumes mounted with the /SYSTEM qualifier are safe from the user with the VOLPRO privilege as long as the user does not also have the SYSNAM privilege.

You should exercise extreme caution in granting the VOLPRO privilege. If unqualified users can override volume protection, the operating system and service to others can be disrupted. Such disruptions can include destruction of the database and exposure of confidential information to unauthorized persons.

6.3.35 WORLD Privilege (System)

The WORLD privilege allows the user's process to affect other processes both inside and outside its group by executing the following process control system services:

- Cancel Wakeup ($CANWAK)
- Delete Process ($DELPRC)
- Force Exit ($FORCEX)
- Resume Process ($RESUME)
- Schedule Wakeup ($SCHDWK)
- Set Priority ($SETPRI)
- Suspend Process ($SUSPND)
- Wake ($WAKE)

The user's process is also allowed to examine processes outside its own group by executing the Get Job/Process Information ($GETJPI) system service. The user with the WORLD privilege can issue the DCL commands SET QUEUE, DELETE/ENTRY, STOP/ENTRY, and SET PROCESS for all other processes.

To exercise control over subprocesses that it created or to examine these subprocesses, a process needs no special privilege. To affect or to examine other processes inside its own group, a process needs only the GROUP privilege. But to affect or examine processes outside its own group, a process needs the WORLD privilege. You should normally grant this privilege to any user who needs to affect other processes on a system-wide basis.

6.4 Accounting for the Use of System Resources

For accounting purposes, the VAX/VMS system keeps records of the use of system resources. These records are kept in the accounting log file SYS$MANAGER:ACCOUNTING.DAT. This file is updated whenever

- The system is initialized.
- An accountable process or image terminates.
- A print job is completed.
- A login failure occurs.

In addition, users can send messages to be inserted in the accounting log file.

Privilege is required to suppress the accounting function and thus avoid accounting for the use of system resources. Only a user who has the ACNT privilege can create subprocesses or detached processes in which accounting is disabled. The /NOACCOUNTING qualifier of the DCL command RUN disables all accounting in a created process.

A user with the OPER privilege can selectively disable various kinds of accounting throughout the system by using the /DISABLE qualifier of the DCL command SET ACCOUNTING. Usually, this action is considered a system management task. See the *VAX/VMS DCL Dictionary* for a full description of the SET ACCOUNTING command.

6.4.1 Accounting Records

Accounting records contain cumulative accounts of the resources used either by processes or images set up for users, or by print symbionts that print out files for users. Each accounting record contains three fields—username, UIC, and account name—that identify the user and establish the connection between the accounting record and a user of the system. These fields correspond to similar fields of the user's account record in the user authorization file (UAF).

As system manager, you can use the Accounting Utility to sort, select, and report the accounting records. The reports can provide valuable system management tools. (See the *VAX/VMS Utilities Reference Volume*.) Alternatively, by using the detailed accounting records provided by the system, you or perhaps a system programmer can devise programs for reporting on the use of system resources and for billing for their use.

6.4.2 Accounting Log File

The accounting log file is created and opened automatically when the operating system is initialized. Accounting records are arranged chronologically in this file. The following list summarizes the characteristics of the accounting log file:

- Filename: ACCOUNTNG.DAT (this file is not an ASCII file; hence, it must be formatted before it is printed)
- Residence and directory: SYS$MANAGER
- File organization: sequential
- Record length: variable
- Record types: eight

The eight types of records correspond to the conditions that cause records to be written to the file. These record types are shown in the list that follows. Note that their corresponding codes, as defined in the macro $ACRDEF in SYS$LIBRARY:STARLET.MLB, are shown in parentheses:

- Records written when processes are deleted (ACR$K_PRCDEL)
- Records written when an image terminates (ACR$K_IMGDEL)

- Records written when the system was initialized (ACR$K_SYSINIT)
- Records written when print jobs are queued (ACR$K_PRINT)
- Records written when login failures occurred (ACR$K_LOGFAIL)
- Records written when users' messages are sent to the accounting log file (ACR$K_USER)
- Records that point to the next accounting file (ACR$K_FILE_FL)
- Records that point to the previous accounting file, if any (ACR$K_FILE_BL)

For more information about the records of the accounting log file, see the *VAX/VMS Utilities Reference Volume*.

As records are entered in the accounting log file, all but image termination records are immediately flushed to disk. This precaution guarantees the integrity of the file and the completeness of accounting data even if the system fails.

Normally, the current version of the accounting log file is closed at the end of a billing period, and a new version is created and opened. As a rule, you perform this job with the SET ACCOUNTING command.

If an attempt to write to the accounting log file results in an error, the file is closed automatically and a new copy is created and opened.

PART II Maintaining the System

7 Maintaining Public Files and Volumes

This chapter contains guidelines for setting up and maintaining public files and volumes. It describes routine setup tasks such as initializing and mounting public volumes. It also describes how to maintain the public environment by performing such tasks as mount verification, volume integrity (BACKUP), and disk space management.

Public volumes, also called system volumes, are file-structured disk volumes that contain public files. Such files must be available to most, if not all, users. Public volumes can also contain files that users create for their own private use or for general use. Thus, as long as UIC-based file protection permits it, all users have access to public volumes and to the files on them.

Public volumes can contain the following kinds of public files supplied by DIGITAL:

- The operating system itself in executable form, and files related to the operating system
- Utility programs in executable form
- Diagnostic and test programs in executable form, and files related to these programs
- Various system libraries such as macro libraries, object module libraries, shared run-time libraries, and error message libraries
- Text files such as help files
- Optional software in executable form, plus related libraries and other files

In addition, you can include on public volumes files that are unique to an installation, which must be accessible to many or all users.

Finally, you can permit any user to create and store files on a public volume. As a rule, you create a default disk file directory on a public volume for each user authorized to use the system (see Chapter 5). Depending on their file protection, user files may be of general or restricted accessibility. Such use of a public volume, however, is subject to limitation: a user is free to create, catalog, and store files on a public volume only if volume protection permits, if the user has write access to a directory on the volume, and if disk quotas permit.

7.1 Setting Up Public File Structures

A major duty in system management is maintaining public file structures. You must strike a balance between your users' needs and the system's available mass storage resources. You must determine how mass storage devices on your system will be configured, which devices will hold public system volumes, which devices will be available for users' private volumes, and how the public volumes will be configured. You are also responsible for creating top-level user file directories as needed, and monitoring users' disk space usage.

7.1.1 Storing User Files

In most circumstances, you will want to dedicate most of your large disk drives to public disk storage. In relatively large system configurations, you should not put user files on the system volume. The system disk will be kept active with paging and swapping, spooling files, maintaining system logs, and so forth. Furthermore, if you ever find it necessary to build a new system disk from scratch, you will find that having user files on it makes the process much more difficult and time-consuming.

If you have a relatively small mass storage configuration, you will have no other choice than to allocate user files on the system volume. Under these circumstances, you should take adequate precautions with disk quotas and user education to ensure that users' files do not exhaust the free space on the system disk.

7.1.2 Using Volume Sets

A volume set is a collection of disk volumes that have been bound into a single entity, by the DCL command MOUNT /BIND. A volume set looks like a single, large volume. Files are automatically allocated anywhere on the volume set that space is available, disk quotas are enforced over the entire set, and a single directory structure covers the whole volume set. If you want to provide a large homogeneous public file space, use a volume set.

If you intend to create database files that are larger than any single disk volume, you must use a volume set. It is worth noting that the file system attempts to balance the load on the volume set, with tactics such as creating new files on the volume that is the least full at the time.

On the other hand, if you want several distinct areas of file storage, with different user bases or different management policies, you must use a separate volume (or volume set) for each area. For example, you might want one volume for permanent user storage, with carefully limited quotas and careful backups; and another volume for "scratch" use, which has liberal or no quotas, is not backed up, and whose files are cleaned out on a periodic basis. Each separate volume or volume set must contain a top-level user file directory for each user who will keep files on that volume.

An advantage of using separate volumes is their modularity. Should one of the drives holding a volume set be out of service, the whole volume set will be unavailable because of the interconnected nature of its directory structure. When a drive holding a single volume is not functioning, only a well-defined set of files becomes unavailable.

For planning purposes, keep in mind the following:

- Any single volume can be turned into a volume set by binding it with a newly initialized volume. Likewise, you can always add another newly initialized volume to an existing volume set.
- You can bind disk volumes of different types into the same volume set.

- You cannot bind two existing separate volumes containing files into a volume set. (The MOUNT command appears to let you do this, but the result will not be a coherent volume set.)
- You need to issue the MOUNT/BIND command only once to effect binding; thereafter, the volume-set association is recorded on the volumes.
- Once you have bound two or more volumes into a volume set, they cannot be separated. The only way to separate a volume set is to selectively copy sets of directories using the Backup Utility (BACKUP).

For more information on volume sets, see the *VAX/VMS DCL Dictionary*.

Caution: DIGITAL recommends that you not make the system disk part of a volume set. While VAX/VMS will continue to boot and run successfully if you do this, optional product installations, maintenance updates, and system upgrades will not install correctly on a system disk that is part of a volume set. Once you have installed an update or software product, system files can be allocated anywhere on the volume set, and VAX/VMS will no longer boot successfully.

7.2 Formatting Disks

Disks that you purchase from DIGITAL are preformatted with the EVRAC disk formatter. However, under some circumstances you may need or want to format a disk. Disks must be reformatted if they have been exposed to X-rays, degaussing, or certain kinds of power disruptions. Also, you may find formatting desirable if you are experiencing excessive parity errors on a disk. In such cases, you should contact your Field Service representative for assistance.

7.3 Initializing Public Volumes

The purpose of initializing a disk volume is to delete all old information from the volume and to impart to the volume a Files–11 structure that the operating system recognizes. This structure prepares a volume to receive data and stores it so that the operating system can locate it easily.

When initializing a public volume, you must specify the /SYSTEM qualifier for the DCL command INITIALIZE. You may also need to use one or more of the following INITIALIZE qualifiers when initializing a public volume:

- /ACCESSED=n
- /CLUSTER_SIZE=n
- /EXTENSION=n
- /HEADERS=n
- /INDEX=position
- /MAXIMUM_FILES=n
- /WINDOW=n

As described below, selecting appropriate values for n and selecting the appropriate position for the /INDEX qualifier often involve making trade-offs. The following guidelines for initializing public volumes supplement information presented in the *VAX/VMS DCL Dictionary.*

/ACCESSED Qualifier

The /ACCESSED qualifier provides an estimate of the number of directories expected to be in use concurrently on a volume. The file system keeps this number of directory file control blocks in system space for ready access depending on which directories were most recently used. The result is a substantial reduction of overhead in directory operations. For volumes mounted with the /SYSTEM qualifier, the system parameter ACP_DINDXCACHE overrides this value.

When you create a volume set, specify reasonable values for the /ACCESSED qualifier on each volume, because the total number of directory file control blocks retained will be the sum of the values of all the /ACCESSED qualifiers specified for the volume set.

/CLUSTER_SIZE Qualifier

The /CLUSTER_SIZE qualifier specifies the fundamental unit of allocation (expressed in blocks) on a volume. In selecting the cluster size, you trade off wasted space at the end of files against the size of the volume storage bit map, which must contain one bit for each cluster on the volume (or one block for each 4096 clusters).

/EXTENSION Qualifier

The /EXTENSION qualifier specifies the default number of blocks allocated for extending files on a volume. This value is less important on the VAX/VMS system than on the RSX-11M, RSX-11D, RSX-11M-PLUS, and IAS systems, because VAX Record Management Services (RMS) uses an adaptive algorithm maximized against /EXTENSION. The value of this qualifier should be an even multiple of /CLUSTER_SIZE.

/HEADERS Qualifier

The /HEADERS qualifier specifies the number of file headers to be allocated initially to the index file. The primary advantage of preallocating file headers is that they will then be located near the storage map file (usually in the middle of the disk). This placement of file headers helps reduce head motion during file manipulation. This value should be estimated conservatively, because space allocated to headers cannot later be made available for file storage.

/INDEX Qualifier

The /INDEX qualifier specifies the location of the index file on a volume. The default position (MIDDLE) results in minimum head motion during file processing. The position BEGINNING should be used if the disk is to contain only one or a few very large contiguous files.

When the Backup Utility copies a volume as the result of a BACKUP/IMAGE command, it preserves the placement of the index file, if the output device is the same type. Otherwise, it defaults to MIDDLE.

/MAXIMUM_FILES Qualifier

The /MAXIMUM_FILES qualifier specifies the maximum number of files that a volume can contain. The default value is fairly liberal. A closer estimate of it helps optimize the dynamic allocation of the index file; once set, however, the maximum number of files for a volume cannot be increased. Note that each directory and each extension header of a multiheader file counts as a file against this maximum value.

/WINDOW Qualifier

The /WINDOW qualifier specifies the default number of map pointers in a file access window. This value is the number of extents of a file to which access can be gained without the cost of file system overhead.

7.4 Mounting Public Volumes

The purpose of mounting a volume or volume set is to establish a relationship between the volume or volume set, the device(s) on which the volume is physically mounted, and one or more processes that can gain access to the volume.

In mounting a public volume (by using the /SYSTEM qualifier to the DCL command MOUNT), you may need to use one or more of the qualifiers /ACCESSED, /EXTENSION, and /WINDOW (described in Section 7.3), or the following additional qualifiers:

- /ASSIST
- /COMMENT=text
- /MOUNT_VERIFICATION
- /PROCESSOR=option

The following guidelines supplement information presented in the *VAX/VMS DCL Dictionary*.

/ASSIST Qualifier

The /ASSIST qualfier enables or disables the operator-assisted mount feature. By default, this feature is enabled. You should encourage your users to take advantage of this feature.

There is a situation where operator-assisted mounts generally prove undesirable. During the execution of SYSTARTUP.COM, operator-assisted mounts are disabled by default. The reason is that the absence of some system volume normally mounted during SYSTARTUP.COM would prevent the system from booting.

If you have one or more volumes that must be present for the correct operation of your system, you can specify the /ASSIST qualifier in the commands to mount them; be prepared for the following consequences, however, if any volume proves to be offline or unavailable.

- The operator assistance software will issue an operator request to have the volume made ready, and will wait for its completion. There is no problem if you can ready the volume.
- If you cannot ready the volume (perhaps the drive is down or the volume is corrupted), you cannot complete SYSTARTUP.COM. To abort the mount request, you would have to issue a REPLY/ABORT operator command. However, the system console is running SYSTARTUP.COM and is unavailable. Furthermore, the other system terminals are usually not yet properly configured or logins are not yet enabled.

Under these circumstances, the only way to boot the system is to invoke the command procedure SYS$SYSTEM:STARTUP.MIN.

/COMMENT Qualifier

The /COMMENT qualifier includes a quoted text string that you specify as part of the mount request. This qualifier is primarily useful in situations where operator assistance is expected, for it passes on information such as the physical location of a particular volume that is required. You should encourage your users to take advantage of this feature.

/MOUNT_VERIFICATION Qualifier

The /MOUNT_VERIFICATION qualifier enables or disables the mount verification feature on Files–11 disks. This feature is explained in Section 7.6. By default, the mount verification feature is enabled. However, note that this feature has no effect on foreign disks or magnetic tapes.

/PROCESSOR Qualifier

The /PROCESSOR qualifier specifies the number of ancillary control processes (ACPs) to be used in controlling various public volumes. Selecting an appropriate option for the /PROCESSOR qualifier involves making a trade-off. If you specify the option SAME, file system parallelism and performance may be sacrificed for the sake of saving system space. Conversely, if you specify the option UNIQUE, system space is sacrificed for the sake of file system parallelism and performance. Unless you have substantial memory to waste, you should use the system default. (See the discussion of the System Generation Utility (SYSGEN) in the *VAX/VMS Utilities Reference Volume* for a description of the ACP_MULTIPLE system parameter.)

You can monitor Files–11 ACP performance with the MONITOR FCP command of the Monitor Utility. (See the *VAX/VMS Utilities Reference Volume.*) The information MONITOR provides can help you determine how to configure your file system.

7.4.1 Preparing Disk and Magnetic Tape Volumes for Use

Users may prepare their own volumes for use. Or, depending on the physical arrangement of the installation and the type of volume to be accessed, you or an operator may be called upon to assist them. (In most cases, an operator performs the tasks described in this section and Section 7.4.2.) The following are some of the reasons why users may require assistance:

- The processor and its peripheral devices are off limits to or remotely located from some or all users.
- The magnetic tape file system has requested that a tape volume be mounted.
- A system or public disk needs to be mounted.

Users normally communicate with an operator to gain access to volumes, and the operator performs one or more of the following tasks:

- Allocates the device on which the volume is placed, using the DCL command ALLOCATE.
- Physically mounts the volume on the device. Physically mounting a volume means placing it on a specific drive and starting the drive. For tape drives, the operator loads the tape into the drive and then presses the **LOAD** button to start the drive. For disk drives, the operator places the disk in the disk drive and then presses the **START** or **RUN** button to start the drive.
- Initializes new volumes, using the INITIALIZE command.
- Mounts the volume, using the MOUNT command.

If it is not necessary to initialize the disk, users can issue a single MOUNT command to allocate the device and request assistance as necessary. For example, a user might enter the DCL command MOUNT to issue the mount request; or MOUNT/COMMENT to also send to the operator's terminal a message specifying the name of the device allocated, the volume to be mounted, the physical identification on the outside of the volume, and its location. (For a complete description of the MOUNT command and qualifiers, refer to the description of the Mount Utility (MOUNT) in the *VAX/VMS Utilities Reference Volume.*) The user then waits until the system responds with a message indicating that the volume is mounted.

Examples of allocating devices and initializing and mounting volumes are provided in the *VAX/VMS DCL Dictionary* under the ALLOCATE, INITIALIZE, and MOUNT commands.

For more information on handling magnetic tape volumes, see the *Guide to VAX/VMS Disk and Magnetic Tape Operations.* You should also consult that manual for information on requests and notifications concerning the mounting and dismounting of disk and magnetic tape volumes.

7.4.2 Responding to Mount Requests from Users

When a user requests that a specific disk or magnetic tape be mounted on a device, the following message format is displayed by the operator communication process (OPCOM) on the operator terminal:

```
%%%%%%%%%%%  OPCOM, dd:mmm:yyyy:hh:mm:ss:cc  %%%%%%%%%%%
request 'request-id,' from user 'USERNAME'
```

For example, a user requesting that the volume TEST_FILES be mounted on the device DMA2: could issue the following command:

```
$ MOUNT DMA2: TEST_FILES/COMMENT=" Shelf slot 6B"
```

This command notifies the operator by displaying at the operator terminal a message similar to the following:

```
%%%%%%%%%%%  OPCOM, 15-APR-1984 15:47:50.26  %%%%%%%%%%%
request 5, from user MALCOLM
Please mount volume TEST_FILES in device _DMA2:
Shelf slot 6B
```

The user receives a message indicating that the operator has received the request:

```
%MOUNT-I-OPRQST, Please mount volume TEST_FILES in device _DMA2:
Shelf slot 6B
```

Besides requesting a specific device for mounting a volume, the user can make a generic MOUNT request by specifying a device type. To mount the volume CITIES on a magnetic tape drive whose name begins with MT, for example, the user would issue the following command:

```
$ MOUNT MT:  CITIES/COMMENT="Slot 12c"
```

If the user has already allocated a drive whose name begins with MT, the Mount Utility will request that CITIES be mounted on that particular drive. If the user has not allocated a drive whose name begins with MT, the utility will allocate the first available TE16, TU45, or TU77 tape drive it finds, and issue a request to mount CITIES on that drive.

To respond to user requests for mounting volumes, the operator proceeds as follows:

1. Places the volume on the specified device. After the operator has located the physical volume and placed it on the device, the user receives messages similar to the following:

   ```
   %MOUNT-I-MOUNTED, TEST_FILES mounted on _DMA2:
   %MOUNT-I-RQSTDON, operator request cancelled.
   ```

2. Performs the necessary startup procedure. On disk drives, the startup procedure requires pressing the **START** or **RUN** button; on magnetic tape drives, it requires pressing the **LOAD** button.

If the user intends to write on a magnetic tape volume to be mounted, the operator verifies that the magnetic tape contains a *write ring* before loading the volume on the drive. A volume that does not contain a write ring is write locked and thus cannot be accessed for write operations.

If the user wants a particular disk to be checked for bad blocks, the operator verifies that the volume has been mounted with the /FOREIGN qualifier. For information on how to invoke and use the Bad Block Locator Utility (BAD), see the *VAX/VMS Utilities Reference Volume*.

For more information on the user's role in mounting volumes, see the *Guide to VAX/VMS Disk and Magnetic Tape Operations*.

7.5 Maintaining Volume Integrity

To enhance performance, the system caches in memory information concerning a volume's free space, file identifications, quota file entries, and file headers. You determine the degree of caching with the ACP cache system parameters (see the discussion of SYSGEN in the *VAX/VMS Utilities Reference Volume*). Individual users can alter cache sizes on their volumes with qualifiers to the DCL command MOUNT (see the *VAX/VMS DCL Dictionary*). The system writes the information in the caches to the disk when the disk is dismounted or the system is shut down. Naturally, removal of a disk before the caches are written back loses any changes made to the information in the caches. Therefore, neither you nor the individual user should

- Write lock a volume while it is mounted
- Remove a volume from a drive until it has been dismounted
- Halt the system without performing an orderly shutdown procedure (see Chapter 4)

If anyone write locks a volume at mount time, the system additionally applies a software write lock. If you need to write enable a volume that was mounted while the **WRITE LOCK** switch was on, you must first dismount the volume, then write enable the drive, and then remount the volume. If a volume was mounted on a drive with write lock off and then someone toggles the **WRITE LOCK** switch on (and if mount verification is enabled for the volume, which it is by default), the volume enters mount verification. All I/O operations to the volume are suspended. Section 7.6.2 describes how recovery is effected with write-lock mount verification. (Without the mount verification facility, you would have to dismount the volume, write enable the drive, and then remount the volume.)

At mount time, if the system detects that the caches were not written back the last time the volume was used, the system automatically rebuilds the file information by scanning the contents of the volume. However, file headers for files open at the time of the improper dismount may be partially or wholly lost.

7.6 Mount Verification

The mount verification feature of Files–11 disk handling generally leaves users unaware that a mounted disk has gone offline and returned online or in some other way has become unreachable and then restored. Mount verification is enabled by default with the /MOUNT_VERIFICATION qualifier when the disk is mounted. To disable mount verification, the user must specify /NOMOUNT_VERIFICATION when mounting the disk. Note that this feature does not apply to foreign disks or to magnetic tapes.

Mount verification sends messages to OPCOM. Because there are cases where mount verification messages are needed at the operator's console and OPCOM might not be able to provide them, mount verification also sends special messages with the prefix %SYSTEM-I-MOUNTVER to the operator's console only, that is, to OPA0. For example, if the system disk undergoes

a mount verification or if OPCOM is not present on a system, the operator would at least receive the messages with the %SYSTEM-I-MOUNTVER prefix. Under normal circumstances, both messages are received at the operator's terminal, with the %SYSTEM-I-MOUNTVER message arriving first.

7.6.1 Device Offline Mount Verification

Mount verification is initiated under the following conditions:

- A disk volume is taken offline (for example, it might be spun down) because of a hardware or user error. Most disk drives generate a special interrupt when the volume comes back online, which causes the software to mark the volume as invalid. However, some disk drives (such as the RX01, RX02, RL02, and TU58) do not generate online attention interrupts. For these devices, an offline condition is only detected when an I/O operation is initiated for the drive.
- An I/O operation fails with a medium offline or volume invalid status. The software marks the volume to indicate that it is undergoing mount verification, and all I/O operations to the disk are stalled.

OPCOM then issues a message to the operators enabled for DISKS and DEVICES to announce the disk's unavailability, in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' is offline.
Mount verification in progress.
```

If a mounted disk volume goes offline while mount verification is enabled, you can act as follows to resume operations:

1. Take the disk out of the offline and verification pending state by shutting down mount verification with one of the three techniques described in Section 7.6.3. These techniques cause the pending and future I/Os to the volume to fail.

2. If the disk drive is faulty, but another functioning drive is available on the same controller, you can move the disk to the functioning drive and swap the unit select plugs.

When the disk comes back online, and is detected by the mount verification software that polls the disk drive, the system verifies that the disk's home block matches the one in the database of mounted volumes, thus confirming that this is the same disk as previously mounted.

If the drive does not contain the correct volume, OPCOM issues a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' contains the wrong volume.
Mount verification in progress.
```

After the mount verification succeeds, the disk is marked as valid. OPCOM issues the following message:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification completed for device 'device-name.'
```

At this point I/O operations to the disk are allowed to proceed, as shown in the following example:

```
%%%%%%%%%%% OPCOM, 15-APR-1984 11:54:54.12 %%%%%%%%%%%
Device DMA0: is offline.
Mount verification in progress.
%%%%%%%%%%% OPCOM, 15-APR-1984 11:57:34.22 %%%%%%%%%%%
Mount verification completed for device DMA0:.
```

In this example, the message from OPCOM informs the operator that device DMA0: has gone offline and mount verification has been initiated. The operator finds that the drive was accidentally spun down and successfully spins it back up. The next message indicates that mount verification is satisfied that the same volume is on the drive (which it has found is online again), and all I/Os to the volume resume.

7.6.2 Device Write-Lock Mount Verification

Devices may become write locked under the following conditions:

- A hardware or user error occurs while a Files–11 disk volume is mounted for writing (the /WRITE qualifier is the default).
- The software discovers that the disk is write locked (typically an I/O fails with a write-lock error). The disk is marked that verification is in progress.

OPCOM issues a message to the operators enabled for DISKS and DEVICES to announce the disk's unavailability:

```
%%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' has been write locked.
Mount verification in progress.
```

If for some reason a mounted disk volume becomes write locked while mount verification is enabled, you can act as follows to resume operations:

1. Take the disk out of the verification pending state by shutting down mount verification with one of the techniques described in Section 7.6.3. These techniques cause the pending and future I/Os to the volume to fail.

2. Enable the drive for writing by toggling the drive's hardware **WRITE LOCK** switch.

3. If the disk drive is faulty, but another functioning drive is available on the same controller, you can move the disk to the functioning drive and swap the unit select plugs. (Note that switching to another drive will cause the volume to undergo offline mount verification. Once that completes, the write-lock mount verification continues.)

When the mount verification software that polls the disk drive determines that the volume is in a writeable state, I/O operations to the disk are allowed to proceed. However, OPCOM does not issue a message indicating that the write-lock mount verification has completed. Instead, OPCOM issues a message in the following format to alert the operator that the device is write locked:

```
%%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Device 'device-name' has been write locked.
Mount verification in progress.
```

You can toggle the **WRITE LOCK** switch on the drive to eliminate the write lock condition. I/O operations to the disk resume, with no further messages.

7.6.3 Canceling Mount Verification

You can cancel a mount verification request in one of three ways:

1. Allow the mount verification in progress to continue the number of seconds defined by the system parameter MVTIMEOUT. When the time expires, the system automatically cancels the pending mount verification. Note that a mount verification initiated by a write lock will not time out.

2. Invoke a special canceling routine from the console terminal.

3. Dismount the volume with the DCL command DISMOUNT from a process that is not hung.

These methods are discussed in the following sections.

7.6.3.1 MVTIMEOUT System Parameter

The MVTIMEOUT system parameter defines the time (in seconds) that is allowed for a pending mount verification to complete before it is automatically canceled (see the discussion of SYSGEN in the *VAX/VMS Utilities Reference Volume*). This dynamic parameter should always be set to a reasonable value for the typical operations at your site. (See Chapter 11 for instructions on how to display and modify dynamic system parameters with SYSGEN). Note that resetting the value of the MVTIMEOUT parameter will not affect a mount verification that is currently in progress.

You will probably find that 10 minutes (600 seconds) is a good value for MVTIMEOUT, whether you normally operate with or without an operator.

When a pending mount verification is canceled by timing out, OPCOM prints a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification aborted for device 'device-name'.
```

After a mount verification times out, all pending and future I/O requests to the volume will fail. Thus, the disk must be dismounted and remounted before it can be accessed again.

Note that a write-lock mount verification will not time out.

7.6.3.2 Cancellation Commands

If you must stop a mount verification before the time specified by MVTIMEOUT can elapse, enter the following sequence of commands from the console terminal of your VAX processor:

```
CTRL/P
>>>HALT
>>>D/I 14 C
>>>CONT
IPC>
```

While you are at the IPC> prompt, all system operation is suspended. (Press CTRL/Z when you are ready to exit.)

Note the following special characteristics of the mount verification canceling routine:

- Lowercase characters are converted to uppercase.
- Illegal characters (such as most control characters) are not echoed; instead, the terminal bell character is issued as a warning alarm.
- Leading spaces are ignored and are not echoed.
- Multiple spaces are compressed into a single space; that is, all space characters after the first are ignored.

There are two commands you can enter in response to the IPC> prompt:

```
IPC> device-name
```

This command cancels any pending mount verification on the device specified. (A warning is given if no mount verification was in progress for that device.)

```
IPC> X
```

This command transfers control to the debugging tool XDELTA (provided it was loaded with the system by setting the appropriate value in the bootstrap file). If XDELTA has not been loaded, the prompt IPC> is reissued. XDELTA, which is described in the *VAX/VMS Utilities Reference Volume*, may prove especially useful if you are debugging privileged software on a VAX–11/782 attached processor system.

Press CTRL/Z to exit from the mount verification canceling routine and resume system operation.

When a pending mount verification is canceled, OPCOM prints a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
Mount verification aborted for device 'device-name.'
```

After you successfully cancel a pending mount verification with this technique, you must dismount and then remount the volume before you can access it again, as shown in the following example:

```
%%%%%%%%%%% OPCOM, 15-JUN-1984 10:54:54.12 %%%%%%%%%%%
        Device DBA0: is offline.
        Mount verification in progress.

CTRL/P
>>>HALT
>>>D/I 14 C
>>>CONT
IPC>C DBA0
IPC>^Z
%SYSTEM-I-MOUNTVER, _DBA0: has aborted mount verification.
%%%%%%%%%%% OPCOM, 15-JUN-1984 10:56:26.13 %%%%%%%%%%%
Mount verification aborted for device DBA0:
```

In this example, you observe that device DBA0: is offline, but are unable to spin the disk back up. There is no other available drive on the controller, so it is not possible to switch the unit select plugs of the two drives. You also reject the possibility of issuing a DISMOUNT command for the disk, because it was mounted as a private volume.

Rather than wait ten minutes for the mount verification to time out, you can invoke the cancellation commands at the console terminal. Observe that the %SYSTEM-I-MOUNTVER message also appears here because this is the console terminal.

7.6.3.3 Dismounting the Volume

In some cases, you can abort mount verification by dismounting the volume in question. This only works if it is possible for you to issue the DCL command DISMOUNT for the volume (for example, if the system file ACP is not hung).

To do so, follow these steps:

1. Log in at another terminal or use any logged-in terminal that has access to the volume.
2. Enter the DISMOUNT command for the volume.

3 When you cancel a pending mount verification by dismounting the volume, OPCOM issues a message in the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc. %%%%%%%%%%
Mount verification aborted for device 'device-name.'
```

If you do not have access to the volume, you will receive an error message. You can try again if you can find an appropriate process to use. If your process hangs, it is the system file ACP that is hung, and you cannot use this technique to cancel mount verification.

4 Once the cancellation succeeds, remove the volume from the drive.

7.7 Backing Up Public Volumes

This section provides system management procedures and considerations for backing up public files and volumes. Backing up a volume means copying the contents of the volume to another volume or set of volumes (for example, another disk or magnetic tape). It is a precautionary measure to allow you to recover from the loss or destruction of valuable information. Most sites establish a policy and a schedule for regularly backing up files on public volumes.

It is just as desirable to back up private volumes as public. However, responsibility for backing up the files on private volumes usually is left to the individual owners of those files and volumes. The *Guide to VAX/VMS Disk and Magnetic Tape Operations* provides a comprehensive introduction to the use of the Backup Utility, including detailed information about backup media and tasks.

There are two kinds of backups for public disk files and volumes:

- Incremental (or partial). Rather than save all the files on a volume every time a save is performed, it is better to save only those files that were created or modified since the last save operation. This type of selective backup is termed an *incremental backup*. Incremental backups are described in Section 7.7.4.

- Full (or all-inclusive).

The backup medium, in either case, can be disk or magnetic tape. Incremental backups efficiently capture only those files that have been modified since the last full backup. Periodic full backups are necessary to provide the basis for reconstruction of a lost volume.

As a rule, incremental backups are performed more frequently than system backups. You should decide, after consulting with users of the system, how frequently to back up files and volumes and how long to retain backup media. Generally, you are responsible for setting up a schedule for backing up files and volumes and for maintaining this schedule.

The following schedule for backing up public disk volumes on magnetic tape affords adequate protection of data for many installations:

- Daily—An incremental backup retained for 7 days. This schedule requires 7 daily magnetic tapes that are rotated once a week.
- Weekly—An incremental backup retained for 4 weeks. This schedule requires 4 weekly sets of magnetic tapes that are rotated once every 4 weeks.
- Monthly—An all-inclusive backup retained for a year. This schedule requires 12 monthly sets of magnetic tapes that are rotated once a year.

Despite all precautions, there is always the risk of losing a file. Frequent backups and longer retention periods reduce this risk.

You can perform full backups either to magnetic tape or to another disk. Each has its advantages and disadvantages. The advantage of using magnetic tape for backups is the much lower media cost. The lower cost may in turn permit you to retain backups longer than you would if you were keeping full backups on disk.

There are several advantages to keeping backups on disk, which in some cases outweigh the higher cost of disk media:

- Disks exhibit better data reliability.
- Disks tend to degrade less in storage.

- If you have to replace a lost volume from its backup medium, a disk backup volume can be ready for immediate use, whereas you must first restore a magnetic tape to disk.
- If you use disks for your backups, you can make use of a "rotating backup set," in which several disks or sets of disks are used in rotation on the system. At the end of each period of use (for example, once a month), the volume or volume set currently in use is copied to the oldest set of disks, the current volume is retired, and the new copy is put online for use during the next period.

The remainder of this section describes backup tasks that are oriented toward system management; it is intended as a guide to aid system managers in maintaining the integrity of public files and volumes in a typical operating system environment. It does not, however, include stand-alone BACKUP. For specific instructions on running stand-alone BACKUP (including steps for backing up and restoring the system disk), see Chapter 2.

You should refer to the *VAX/VMS Utilities Reference Volume* for complete information on the VAX/VMS Backup Utility and its qualifiers. For a detailed approach to the use of BACKUP media and tasks, refer to the *Guide to VAX/VMS Disk and Magnetic Tape Operations.*

7.7.1 Rotating Backup Sets

A rotating backup set offers two major advantages:

1. Your backup copy (the volume or volume set just retired) is known to be good, since it has been in use. The integrity of the new copy will be confirmed by its subsequent use; any defects discovered can be repaired using the backup copy.
2. The free space on the new volume is compressed, and all the files on it are made contiguous or almost contiguous, resulting in better file system performance.

There are four disadvantages to a rotating backup set:

1. Rotating backup sets are more vulnerable to disk errors than sets created by retiring the copy and continuing to run with the original. A disk error during the copy operation results in corrupted data on your new volume; disk errors in directories or file headers will result in the loss of one or

more files. Thus, you must monitor the copy operation very carefully for errors and manually repair any problems that arise.

2 You cannot perform the copy operation while users are updating files. The volume or volume set must be write locked so that the copy will be consistent.

3 Files created with explicit placement lose their placement when the volume is copied. You should therefore not use a rotating backup set if the volume's primary contents are a set of database files that were carefully placed for optimized performance.

4 Rotating backup sets are expensive to apply to fixed-media disks (as opposed to removable media).

7.7.2 Backing Up Full Volumes and Volume Sets

If you want to back up an entire disk volume or volume set, you should use the /IMAGE qualifier of the Backup Utility. You can perform an image backup in a copy operation (disk to disk) or in a save operation (to disk or magnetic tape). When you use the /IMAGE qualifier in a save operation, BACKUP creates a save set that contains data necessary for reinitializing the disk volume. (The attribute record that contains this information is called the save-volume summary record.)

When you use the /IMAGE qualifier in a copy or restore operation, the output volume is reinitialized; the restored volume is a functionally equivalent copy of the original volume. If the output volume has already been initialized as a Files–11 volume, you can use the /NOINITIALIZE qualifier to reinitialize the volume using the volume initialization parameters found on the volume.

You cannot use other file selection qualifiers with the /IMAGE command qualifier. All files on the disk are saved, including reserved files and lost files (files that have no directory entry).

You should specify the /RECORD command qualifier if you are performing the full volume backup in conjunction with incremental backups that use the /RECORD qualifier.

You can back up an entire volume set by following the same procedures outlined in the next section for backing up a disk volume. Simply name the device on which the root volume (volume number 1) is mounted.

7.7.2.1 Backing Up a Public Disk to Disk

The procedure below describes how to copy the contents of a public disk to another disk. (A public disk is a disk that has been mounted with the /SYSTEM qualifier.)

You should make certain that your target volume has sufficient free space. You need only write lock the source volume device to prevent users from changing any data on the disk. However, users still can read data.

1 Issue the following command to warn all users that the disk will be dismounted and write locked for backup:

```
$ REPLY/ALL/BELL "message-text"
```

This message should include the name of the source disk being write locked and indicate in how many minutes the write lock will occur.

2 Use the SHOW DEVICE/FILE command to ensure that all files on the disk have been closed; then broadcast messages to users who have open files, as necessary.

3 At the time indicated by the message, issue the DISMOUNT /NOUNLOAD command to logically dismount the source disk as follows:

```
$ DISMOUNT/NOUNLOAD device-name:
```

4 Write lock the source disk by pressing the **WRITE PROTECT** switch to the ON position. This switch is located on the front panel of the disk drive.

5 Mount the source disk again as follows:

```
$ MOUNT/SYSTEM device-name: volume-label
```

6 Place the target disk in the drive and ready the device by pressing the **RUN/STOP** button or the **START/STOP** switch.

7 Mount the target volume as follows:

```
$ MOUNT/FOREIGN device-name:
```

8 Invoke the Backup Utility as follows:

```
$ BACKUP/IMAGE/VERIFY input-device:  output-device:
```

If BACKUP returns any verification error messages, refer to the *VAX/VMS System Messages and Recovery Procedures Reference Manual.*

9 Dismount the target volume with the following command:

```
$ DISMOUNT device-name:
```

10 Remove the target volume from the drive and place a label on the outside of the volume; the label should include the volume label and current date.

11 Dismount the source disk by issuing the DISMOUNT /NOUNLOAD command as follows:

```
$ DISMOUNT/NOUNLOAD device-name:
```

12 Write enable the source disk by pressing the **WRITE PROTECT** switch to the OFF position.

13 Remount the source disk with the MOUNT command as follows:

```
$ MOUNT/SYSTEM device-name: volume-label
```

14 Inform all users that the source disk is no longer write locked by issuing the following command:

```
$ REPLY/ALL/BELL "message-text"
```

The following example illustrates these procedures:

```
$ REPLY/ALL/BELL "DMA2:  WILL BE WRITE LOCKED IN 5 MINS. FOR BACKUP."
_OPA0:.SYSTEM    06:31:29.78
DMA2:  WILL BE WRITE LOCKED IN 5 MINS. FOR BACKUP.
$ SHOW DEVICE/FILES DMA2:
   .
   .
   .
$ DISMOUNT/NOUNLOAD DMA2:
$ MOUNT/SYSTEM DMA2: PUBLIC
%MOUNT-I-WRITELOCK, volume is write locked
%MOUNT-I-MOUNTED, PUBLIC     mounted on _DMA2:
$ MOUNT/FOREIGN DMA1:
%MOUNT-I-MOUNTED, PUBLIC     mounted on _DMA1:
$ BACKUP/IMAGE/VERIFY DMA2: DMA1:
$ DISMOUNT DMA1:
$ DISMOUNT/NOUNLOAD DMA2:
$ MOUNT/SYSTEM DMA2: PUBLIC
%MOUNT-I-MOUNTED, PUBLIC    mounted on _DMA2:
$ REPLY/ALL/BELL "DMA2: IS NO LONGER WRITE LOCKED."
_OPA0:.SYSTEM    06:46:44.23
DMA2: IS NO LONGER WRITE LOCKED.
```

You can use BACKUP either to copy files to the new disk or to create a save set on the new disk. If you create a save set on the new disk, you must first create a directory to which the save set will be written and you must use the /SAVE_SET output qualifier. The directory must be included in the save-set name or it must be the default directory.

For example:

```
$ SHOW DEFAULT
DMA1:[SYSTEM]
$ BACKUP/IMAGE DMA1: DB4:[BACKUPS]21JANDMA1.BCK/SAVE_SET
```

7.7.2.2 Backing Up a Public Disk to Magnetic Tape

If you use magnetic tape as the backup medium, you may need to mount additional magnetic tapes. The number depends on the size of the disk being saved and the amount of space in use on the disk.

An image backup operation to magnetic tape always creates a save set; therefore, you need not include the /SAVE_SET output qualifier. To perform the operation, you should follow the steps given in Section 7.7.2.1 for backing up to disk. However, if you are using a fresh magnetic tape, you should initialize it by including the /REWIND qualifier with the BACKUP command.

For example:

```
$ BACKUP/IMAGE/VERIFY/REWIND DRA1:  MTA0:1JUN.BCK
```

7.7.2.3 Backing Up a Disk Volume Set When Drives Are Limited

When you use BACKUP with a volume set, you must mount all volumes of the set. Therefore, it may not be possible to copy a large volume set directly to disk if you have a limited number of disk drives. You can copy the volume set one volume at a time with the BACKUP/IMAGE/VOLUME command, using one more drive than the number of volumes in the volume set. You must write lock the volume set during the entire procedure to ensure consistency.

For example:

```
(suitable warning broadcasts)
    .
    .
    .
$ DISMOUNT/NOUNLOAD DRA0:
$ MOUNT/SYSTEM/NOWRITE DRA0:,DRA1:,DRA2: PUBLIC01,PUBLIC02,PUBLIC03
%MOUNT-I-MOUNTED, PUBLIC01 mounted on _DRA0:
%MOUNT-I-MOUNTED, PUBLIC02 mounted on _DRA1:
%MOUNT-I-MOUNTED, PUBLIC03 mounted on _DRA2:
$ MOUNT/FOREIGN DRA3:
%MOUNT-I-MOUNTED, SCRATCH01 mounted on _DRA3:
$ BACKUP/IMAGE/VOLUME=1 DRA0: DRA3:
$ DISMOUNT DRA3:
$ MOUNT/FOREIGN DRA3:
%MOUNT-I-MOUNTED, SCRATCH02 mounted on _DRA3:
$ BACKUP/IMAGE/VOLUME=2 DRA0: DRA3:
$ DISMOUNT DRA3:
$ MOUNT/FOREIGN DRA3:
%MOUNT-I-MOUNTED, SCRATCH03 mounted on _DRA3:
$ BACKUP/IMAGE/VOLUME=3 DRA0: DRA3:
$ DISMOUNT DRA3:
$ DISMOUNT/NOUNLOAD DRA0:
$ MOUNT/SYSTEM DRA0:,DRA1:,DRA2: PUBLIC01,PUBLIC02,PUBLIC03
%MOUNT-I-MOUNTED, PUBLIC01 mounted on _DRA0:
%MOUNT-I-MOUNTED, PUBLIC02 mounted on _DRA1:
%MOUNT-I-MOUNTED, PUBLIC03 mounted on _DRA2:
    .
    .
    .
(announce public disk is available again)
```

In this example, the operator needs to back up a three-volume set. Thus, at least four drives are required. The operator warns the users that drives DRA0:, DRA1:, and DRA2: are about to be write locked for backups. Next the operator issues MOUNT commands for the volumes PUBLIC01, PUBLIC02, and

PUBLIC03 on drives DRA0:, DRA1:, and DRA2:, respectively, to specify write locking with the /NOWRITE qualifier. Note that PUBLIC01 is the root volume. A scratch volume is mounted on drive DRA3: to be used for the target volumes. As the volume on DRA3: is filled, it is dismounted and a new scratch volume is mounted. This procedure repeats until the third volume has been copied. Then the last volume on DRA3: is dismounted. To enable writing on the volume set again, the operator dismounts (but does not unload) DRA0:, and then mounts the volumes again, omitting the /NOWRITE qualifier. As a final step, the operator announces to the users that the volume set is available again for writing.

Note: **Dismounting the public volume set, remounting write locked, and so forth, is critical to the success of the operation. *Do not* simply write lock a mounted volume; doing so is likely to cause errors and/or suspension of system activity.**

7.7.2.4 Writing Save Sets on Sequential-Disk Volumes

As an alternative to standard Files–11 structure, you can write save sets to disk sequentially, as they would be written on magnetic tape. The primary advantage in using sequential-disk save sets is that you can treat the disk volume as you would a magnetic tape save set; this flexibility allows you to mount multivolume save sets one volume at a time.

To save data to a sequential-disk save set, make certain that the output specifier includes a device name for a disk device, a save-set name, and the qualifier /SAVE_SET; do not include a directory name.

To write sequential-disk save sets, you must mount the output disk with the /FOREIGN qualifier. Although the volume is mounted foreign, BACKUP manages the disk using Files–11 structure.

Volumes that are to be used for sequential-disk save sets either should be freshly initialized or should contain only save sets. Volumes that have been used for general file processing are usually unsuitable for sequential-disk use. You can initialize the volume by using the BACKUP qualifier /INITIALIZE.

Note that unless you use the /INITIALIZE qualifier, the following restrictions apply to the output volume:

- The disk must be Files–11 Structure Level 2.
- The disk must not be part of a normal Files–11 volume set.
- The cluster factor of the disk must be 1.
- The free space on the disk cannot be fragmented into more than 100 contiguous extents.
- The index file cannot be extended.
- The master file directory (MFD) cannot be extended.

For example, you can use the following commands to mount and initialize a sequential disk on drive DMA0: and save the file CONTRACTS.DAT to a save set named CONTRACTS.BCK:

```
$ MOUNT/FOREIGN DMA0:
$ BACKUP/LOG CONTRACTS.DAT DMA0:CONTRACTS.BCK -
_$ /SAVE_SET/INITIALIZE/LABEL=SAVEWORK
%BACKUP-S-COPIED, copied DBA2:[USER]CONTRACTS.DAT;1
```

The MOUNT/FOREIGN command tells BACKUP to write to the output volume sequentially. The /SAVE_SET qualifier on the BACKUP command output specifier is necessary for writing the output file in BACKUP format. The /INITIALIZE qualifier destroys any previous structure information on the disk, enables the volume to be overwritten, and assigns the volume label SAVEWORK.

When you initialize the volume, the volume label you specify is permanent; that is, the label remains in effect until either of the following takes place:

- You change it with the DCL command SET VOLUME /LABEL.
- The volume is reinitialized.

See the *VAX/VMS Utilities Reference Volume* for more information on BACKUP/INITIALIZE. See the *VAX/VMS DCL Dictionary* for more information on SET VOLUME.

Writing Multivolume Sequential-Disk Save Sets

You may find the multivolume option particularly useful if your system has no tape drive but does have a large fixed-media disk and a small removable disk. When one sequential disk is full, BACKUP prompts you for another disk, as it would for save sets on magnetic tape. You can use more than one disk device at a time to save or restore data; this allows processing to continue on another disk while the one most recently used is spinning down. Note that you need the user privilege LOG_IO to read or write a multivolume sequential-disk save set.

BACKUP normally does not initialize the first sequential-disk volume, as the default is /NOINITIALIZE; however, continuation volumes are always initialized. You can initialize the first volume of a sequential-disk save set by specifying the /INITIALIZE qualifier.

Note: **A set of disks written with BACKUP's sequential-disk handling is referred to as a loosely coupled volume set. That is, it lacks some of the informational structures that are present in a normal volume set, such as the volume set list file. Because of the subtle differences in the structure, you should not write files onto a sequential-disk volume as if it were a normal Files-11 disk; confusing and somewhat obscure errors may result. Because the sequential-disk volumes are part of a volume set, they cannot be processed individually by the Verify Utility.**

7.7.3 Performing Selective Backups

You may find selective backups necessary for certain groups of files that require special treatment. Generally, if files must be backed up regularly, you should create a command procedure that contains the required backup commands. (For more information on creating command procedures, see the *Guide to Using DCL and Command Procedures on VAX/VMS.*)

One way to perform selective backups is through the use of wildcards. You can use the asterisk wildcard (*) to select files by filename, file type, or version number. The command in the following example selects files by version number and copies them to the directory [SAVE] on a specified RK07 drive:

```
$ BACKUP/LOG *.*;3 DMA1:[SAVE]
%BACKUP-S-CREATED, created DMA1:[SAVE]CHARTS.DAT;3
%BACKUP-S-CREATED, created DMA1:[SAVE]FIGURES.DAT;3
%BACKUP-S-CREATED, created DMA1:[SAVE]LISTS.DAT;3
%BACKUP-S-CREATED, created DMA1:[SAVE]TABLES.DAT;3
```

The commands in the next example copy files from a directory and a directory tree on DMA0: to newly created directories on DMA1:. All files in the [GEORGE] directory and its subdirectories on the source disk are copied to directories with the same name on the target volume; the /OWNER_UIC=ORIGINAL qualifier ensures that the output files retain the owner UICs of the input files. All files with the filename DUNGEON in the [SYSEXE] directory on the source disk are copied to the [SYSEXE] directory on the target volume.

```
$ BACKUP DMA0:[GEORGE...]  DMA1:[*...]/OWNER_UIC=ORIGINAL
$ BACKUP DMA0:[SYSEXE]DUNGEON.*;*  DMA1:[SYSEXE]
```

BACKUP has qualifiers that allow you to select files according to criteria such as UIC, creation date, expiration date, and modification date. For example, you can select files according to UIC by using the /OWNER_UIC qualifier as follows:

```
$ BACKUP/LOG [.MEM]/OWNER_UIC=[300,16] $1$DLA2:[USER]
%BACKUP-S-CREATED, created $1$DLA2:[USER]ASSIGN.LIS;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]BBALL.DIS;19
%BACKUP-S-CREATED, created $1$DLA2:[USER]BBLEAGUE.FISTATEMENT;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]BRULES.LIS;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]GYM.MAI;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]INFO.DAT;2
%BACKUP-S-CREATED, created $1$DLA2:[USER]MEMO.MAI;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]NEWS.MAI;1
%BACKUP-S-CREATED, created $1$DLA2:[USER]TEAM.LIS;19
```

In this example, all files residing in the subdirectory [.MEM] with an owner UIC of [300,16] are processed. Note that the brackets are required when you specify the UIC code. You can specify the UIC of the current default process by simply entering the /OWNER_UIC qualifier without a code.

If you want to select your input files according to their date of creation, you can use the /CREATED qualifier with /BEFORE or /SINCE. The following command copies all files in the directory [TRUBBLE] that were created on or since April 3, 1984:

```
$ BACKUP DMA1:[TRUBBLE]*/SINCE=03-APR-1984/CREATED DBA2:[SPRS]
```

You can use the /EXPIRED qualifier with /SINCE or /BEFORE to select files according to the expiration date recorded in the file header:

```
$ BACKUP *.COB;*/BEFORE=20-JUL-1984:/EXPIRED MFA0:JUL20SAVE.BCK
```

In this example, all versions of all files in the current default directory that have a file type of COB and an expiration date earlier than July 20, 1984, will be saved to JUL20SAVE.BCK on MFA0:.

In some cases, you may find it more practical to exclude certain files from a backup operation. You can do this as in the following example:

```
$ BACKUP DBA2:[OSCAR...]/EXCLUDE=(*.LOG;*,NOTES.*;*) MTA1:SAVE.BCK
```

This command directs BACKUP to save all the files in the directory [OSCAR] and its subdirectories, except for those with a file type of LOG and those with the filename NOTES, regardless of file type or version number. The parentheses are required when a list is specified with the /EXCLUDE qualifier.

7.7.4 Performing Incremental Backups

To perform incremental backups, use the /SINCE=BACKUP input qualifier and the /RECORD command qualifier. The /SINCE=BACKUP input qualifier directs BACKUP to select only those files that have been created or modified since the last BACKUP/RECORD operation. The /RECORD qualifier directs BACKUP to record the current date in the backup date field of each file's header.

For example:

```
$ BACKUP/RECORD DB2:[*...]/SINCE=BACKUP MTA0:19OCT.BCK
```

To use the /RECORD command qualifier, you must own the files or have the user privilege SYSPRV.

Note: If you use the /RECORD command qualifier to perform incremental backups on disk volumes, it is a good idea to discourage its use by other users. Incomplete backups may occur if other users perform backups using the /RECORD command qualifier.

The drawback to performing incremental backups is that you accumulate a large number of magnetic tape or disk volumes containing the incremental save sets.

7.7.5 Performing Daily Backups

In performing daily backups, you should follow the suggestions in Section 7.7.4 for incremental backups, but you may want to include the /IGNORE=INTERLOCK qualifier. This qualifier instructs BACKUP to copy files even if they are open for writing; this means that you may produce a copy of a file in its partial state. If you do not use the /IGNORE=INTERLOCK qualifier, an open file will generate an error message and will not be backed up. It will then have to wait for the next incremental backup.

You may find the /IGNORE=INTERLOCK qualifier especially useful if you have files that are always open (and would never be backed up otherwise).

For example:

```
$ MOUNT/FOREIGN MTA0:
%MOUNT-I-MOUNTED, INCD5J mounted on _MTA0:
$ BACKUP/IGNORE=INTERLOCK/RECORD/SINCE=BACKUP -
_$ PUBLIC:[*...] MTA0:INCD12JUN
$ DISMOUNT MTA0:
```

Note that the /IGNORE=INTERLOCK qualifier requires the user privilege SYSPRV, a system UIC, or ownership of the volume.

7.7.6 Performing Weekly Backups

You can perform weekly backups in much the same manner as daily backups. However, the /SINCE qualifier specifies the date of the last weekly backup, normally one week earlier.

The following example assumes that the current date is 14-OCT-1984:

```
$ MOUNT/FOREIGN MTA0:
%MOUNT-I-MOUNTED, INCW7J mounted on _MTA0:
$ BACKUP/IGNORE=INTERLOCK/RECORD/SINCE=7-OCT-1984 -
_$ PUBLIC:[*...]MTA0:INCW14JUN
$ DISMOUNT MTA0:
```

7.7.7 Performing Monthly Backups

Monthly backups should be all-inclusive; that is, they should be full backups. Thus, you specify the /IMAGE qualifier to copy not just the files, but all the information required to initialize the volume or volume set when you need to perform a restore operation.

For example:

```
$ MOUNT/FOREIGN MTAO:
%MOUNT-I-MOUNTED, FULLMA mounted on _MTAO:
$ MOUNT/FOREIGN MTA1:
%MOUNT-I-MOUNTED, FULLO2 mounted on _MTA1:
$ BACKUP/IMAGE/IGNORE=INTERLOCK/RECORD PUBLIC: -
_$ MTAO:FULLJUN82,MTA1
%BACKUP-I-RESUME, resuming operation on volume 2
%BACKUP-I-RESUME, resuming operation on volume 3
%BACKUP-I-RESUME, resuming operation on volume 4
    .
    .
    .
$ DISMOUNT MTAO:
$ DISMOUNT MTA1:
```

7.7.8 BACKUP Media Security

The file system treats a BACKUP save set, whether it is stored on disk or on magnetic tape, as a single file. BACKUP does not check protection on individual files within the save set. Therefore, it is crucial to the system's file security to protect save sets adequately. You should assign save-set files a restrictive file protection. You can use the /OWNER_UIC and /PROTECTION qualifiers to the BACKUP command, as described in the *VAX/VMS Utilities Reference Volume*. Provide physical security for save sets that you keep offline (for example, magnetic tapes and sequential disks); if possible, you should lock them up.

If a user comes to you wanting to restore a particular file, you should not lend out the backup volume because you could give away access to all the files on the volume. The safest way to restore a particular file is for you to mount the volume and restore the file with a command such as the following:

```
$ BACKUP MTAO:SAVESET/SELECT=[JONES.TEXTPROC]LASTMONTH.DAT[*...] -
_$ /OWNER_UIC=ORIGINAL
```

The file will be restored with its original directory, ownership, and protection, allowing the file system to determine whether the user was ever allowed access to the file.

7.7.9 Using BACKUP Journal Files

A BACKUP journal file contains records of BACKUP save operations and the file specifications of the files that were saved with each operation. To find a particular file in a multivolume save set, you can review the BACKUP journal file to find the magnetic tape volume that contains the file.

Use the /JOURNAL qualifier to create, or append information to, a BACKUP journal file. If no file specification appears with the /JOURNAL command qualifier, the name of the BACKUP journal file defaults to SYS$DISK:BACKUP.BJL. The default file type for BACKUP journal files is BJL.

If the specified BACKUP journal file does not exist, it is created; if it already exists, the new journal information is appended to the existing journal file. You can start a new version of a BACKUP journal file by creating an empty file with an editor such as EDT or the DCL command CREATE.

To list a BACKUP journal file, enter a command in this format:

```
$ BACKUP/LIST[=file-spec]/JOURNAL[=file-spec]
```

You must not specify an input- or output-specifier with a BACKUP/JOURNAL/LIST command. If the file specification is omitted from the /LIST qualifier, output is directed to SYS$OUTPUT; if the file specification is omitted from the /JOURNAL qualifier, the default BACKUP journal file is used.

You can use file selection qualifiers with the BACKUP /JOURNAL/LIST command. They allow you to locate a set of files in a save set just as the DIRECTORY command allows you to locate a set of files on a disk. The following example shows all files in the directory [SMITH.PROGS] backed up after October 5, 1984, listed in the BACKUP journal file ARCH.BJL:

```
$ BACKUP/LIST/JOURNAL=ARCH.BJL/SELECT=[SMITH.PROGS]/SINCE=5-OCT-1984
```

The following example shows the use of BACKUP journal files:

```
$ BACKUP/JOURNAL/LOG/IMAGE  DRA2: MTA0:3OCT.FUL
    .
    .
    .
         (first set is printed here)
    .
    .
    .
%BACKUP-I-RESUME, resuming operation on volume 2
%BACKUP-I-READYWRITE, mount volume 2 on _MTA0: for writing
Press return when ready: [RET]
    .
    .
    .
         (second set is printed here)
    .
    .
    .
$ BACKUP/JOURNAL/LIST
Listing of BACKUP journal
Journal file _DB2:[SYSMGR]BACKUP.BJL;1 on 3-OCT-1984 00:40:56.36
Save set 3OCT.FUL created on 3-OCT-1984 00:40:56.36
Volume number 1, volume label 3OCT01
        [COLLINS]ALPHA.DAT;4
        [COLLINS]EDTINI.EDT;5
        [COLLINS]LOGIN.COM;46
        [COLLINS]LOGIN.COM;45
        [COLLINS]MAIL.MAI;1
        [COLLINS]MAR.DIR;1
        [COLLINS.MAR]GETJPI.EXE;9
        [COLLINS.MAR]GETJPI.LIS;14
                  .
                  .
                  .
        [LANE]LES.MAI;1
                  .
                  .
                  .
Save set 3OCT.FUL created on 3-OCT-1984 00:40:56.36
Volume number 2, volume label 3OCT02
        [LANE]MAIL.MAI;1
        [LANE]MEMO.RNO;5
        [LANE]MEMO.RNO;4
                  .
                  .
                  .
        [WALTERS.VI]KD.RNO;52
End of BACKUP journal
```

7.7.10 Backing Up the Console Medium

The procedures for backing up and restoring the console medium are processor-dependent. You can find descriptions of the procedures for your VAX processor in Chapter 2.

7.7.11 Backing Up the System Disk (Using Stand-Alone BACKUP)

To back up and restore the system disk, you should use stand-alone BACKUP as described for your VAX processor in Chapter 2.

7.7.12 Restoring Entire Disk Volumes

The command format for restoring from a save set is as follows:

```
$ BACKUP save-set-specifier[/SAVE_SET] output-specifier
```

You use the /SAVE_SET qualifier when the save-set specifier refers to a disk volume.

To restore an entire disk volume from a save set that was created using the /IMAGE command qualifier, you must mount the new volume using the DCL command MOUNT/FOREIGN. You must then restore the volume with the BACKUP/IMAGE command.

When a save set is created using the /IMAGE command qualifier, the data necessary for reinitializing the volume is placed in the save set. When the /IMAGE command qualifier is used to restore the volume, the new volume is initialized using that data in the save set.

For example:

```
$ MOUNT/FOREIGN DRA1:
%MOUNT-I-MOUNTED, 24JUND mounted on _DRA1:
$ BACKUP/IMAGE MTA0:24JUNDMA1.BCK  DRA1:
```

The volume is initialized using the initialization parameters saved in the save set. All files and directories in the save set are restored to the new volume.

To restore a volume set in image mode, you must mount all the volumes of the set as foreign volumes. You must also, in the output-specifier of the BACKUP command, include the list of devices on which the volumes are mounted.

For example:

```
$ MOUNT/FOREIGN DRAO:
%MOUNT-I-MOUNTED, PUBLIC01 mounted on _DRA0:
$ MOUNT/FOREIGN DRA1:
%MOUNT-I-MOUNTED, PUBLIC02 mounted on _DRA1:
$ MOUNT/FOREIGN DRA2:
%MOUNT-I-MOUNTED, PUBLIC03 mounted on _DRA2:
$ BACKUP/IMAGE MTA0:31JUNPUB DRA0:,DRA1:,DRA2:
```

The volumes receive volume set numbers in the order in which they are listed in the BACKUP command. In this example, DRA0:, DRA1:, and DRA2: become volumes 1, 2, and 3, respectively.

Changing Volume Initialization Parameters Before Restoring

If you wish to change the volume initialization parameters for a volume, you need to perform the following steps:

1 Initialize the new volume using the new initialization parameters.

2 Mount the new volume using the /FOREIGN qualifier.

3 Restore the volume with BACKUP, using the /NOINITIALIZE command qualifier.

In the following example, the cluster size of the volume is being changed to 3. The other volume initialization parameters take the default values from the INITIALIZE command.

```
$ ALLOCATE DRA2:
%DCL-I-ALLOC, _DRA2: allocated
$ INITIALIZE/CLUSTER_SIZE=3 DRA2: TEST_PROGS
$ MOUNT/FOREIGN DRA2:
%MOUNT-I-MOUNTED, TEST_PROGS mounted on _DRA2:
$ BACKUP/IMAGE/NOINITIALIZE/TRUNCATE  MTA0:1JUN.BCK  DRA2:
```

The only initialization parameter that you cannot change is the structure level of the volume. If you change the cluster factor, as in the above example, it is good practice to include the /TRUNCATE qualifier.

7.7.13 Restoring a Volume from Incremental Backups

If you have been performing a combination of full and incremental backups on a public volume, you must use the following procedure to recover the volume from its backups, should the volume be lost.

First, restore the volume from the last full backup, using an image restore operation as shown in the following example (the /RECORD qualifier is required for the correct operation of this procedure):

```
$ MOUNT/FOREIGN DRAO:
%MOUNT-I-MOUNTED, SCRATCH mounted on _DRAO:
$ MOUNT/FOREIGN MTAO:
%MOUNT-I-MOUNTED, FULLMA mounted on _MTAO:
$ MOUNT/FOREIGN MTA1:
%MOUNT-I-MOUNTED, FULL02 mounted on _MTA1:
$ BACKUP/IMAGE/RECORD MTAO:FULLJUN84,MTA1 DRAO:
%BACKUP-I-RESUME, resuming operation on volume 2
%BACKUP-I-RESUME, resuming operation on volume 3
%BACKUP-I-RESUME, resuming operation on volume 4
     .
     .
     .
$ DISMOUNT MTAO:
$ DISMOUNT MTA1:
$ DISMOUNT/NOUNLOAD DRAO:
```

Now mount the disk as a file-structured volume and apply the incremental backups in reverse chronological order. Start with the last daily backup; then apply the preceding daily backups, and finally the weekly backups, as follows:

```
$ MOUNT DRAO: PUBLIC
%MOUNT-I-MOUNTED, PUBLIC mounted on _DRAO:
$ MOUNT/FOREIGN MTAO: INCD17
%MOUNT-I-MOUNTED, INCD17 mounted on _MTAO:
$ BACKUP/INCREMENTAL  MTAO:INCD17JUN DRAO:
$ DISMOUNT MTAO:
    .
    .
    .
$ MOUNT/FOREIGN MTAO: INCD16
%MOUNT-I-MOUNTED, INCD16 mounted on _MTAO:
$ BACKUP/INCREMENTAL  MTAO:INCD16JUN DRAO:
$ DISMOUNT MTAO:
    .
    .
    .
$ MOUNT/FOREIGN MTAO: INCD15
%MOUNT-I-MOUNTED, INCD15 mounted on _MTAO:
$ BACKUP/INCREMENTAL  MTAO:INCD15JUN DRAO:
$ DISMOUNT MTAO:
    .
    .
    .
$ MOUNT/FOREIGN MTAO: INCW14
%MUUNT-I-MOUNTED, INCW14 mounted on _MTAO:
$ BACKUP/INCREMENTAL  MTAO:INCW14JUN DRAO:
$ DISMOUNT MTAO:
    .
    .
    .
$ MOUNT/FOREIGN MTAO: INCW7J
%MOUNT-I-MOUNTED, INCW7J mounted on _MTAO:
$ BACKUP/INCREMENTAL  MTAO:INCW7JUN DRAO:
$ DISMOUNT MTAO:
```

In this example, applying the latest incremental backup using the /INCREMENTAL qualifier causes the volume's directories to be restored to their state at the time the backup was taken. In addition, all the files in the incremental save set are restored. Files that are present on the volume from the full restore operation, but are not present in the directories of the incremental backup, are deleted. (These files were deleted by users during the time period between the full backup and the last incremental backup.)

In applying the earlier incremental backups, BACKUP restores the remaining files that have directory entries on the volume. These are files that were last modified some time before the last incremental backup, and were still present when the last incremental backup was taken.

Note: BACKUP will restore the volume correctly regardless of the order in which the incremental backups are applied; using reverse chronological order is most efficient.

The /RECORD and /INCREMENTAL qualifiers must be used where shown in the above example to obtain the correct operation.

If you choose to selectively exclude certain files in your incremental backups (for example, listing files or batch logs), these files will not be restored, but will have directory entries in the resulting volume. You can clean up these "null" directory entries by running a repair pass with the VAX/VMS Verify Utility (VERIFY). For more information on VERIFY, see the *VAX/VMS Utilities Reference Volume.*

Note that if directory files were renamed during the time period covered by the incremental backups, these directories will appear on the reconstructed volume under both their old and new names. The files that were written since the directory was renamed will appear under the new name; the other files, written before the directory was renamed, appear under the old name. You must merge the old and new directories manually.

7.7.14 Restoring Files and Directories

You can restore individual files in large save sets by using the BACKUP/LIST/JOURNAL command to find the volume that contains the files. You can then mount the volume and use BACKUP to select and restore the desired files. If the volume is not the first volume in a multivolume save set, you will receive the warning message:

```
%BACKUP-W-NOT1STVOL, tape 'name' is not the start of a save set
```

7.7.14.1 Listing the Contents of a BACKUP Save Set

You can use the /LIST command qualifier to list information about files in a save set, either at the terminal or in a specified output file. BACKUP can perform this operation in conjunction with any other BACKUP operation, or by itself.

The following command will list save-set information at the terminal:

```
$ BACKUP/LIST 2MAR1984.BCK/SAVE_SET
```

To list the same information in a file named INFO.LIS, enter the following command:

```
$ BACKUP/LIST=INFO.LIS 2MAR1984.BCK/SAVE_SET
```

If you do not know the save-set names on a magnetic tape volume, you can specify the device alone. Note, however, that BACKUP then performs the list operation as it would a restore operation with only a device specification: It reads the first save set it encounters on the magnetic tape and stops processing when it reaches the end of that save set. BACKUP does not automatically rewind until it reaches the end-of-tape (EOT) (unless you have specified the /REWIND qualifier); therefore, you can proceed to the next save set (if any exists) by repeating the command.

By including an asterisk (*) with the device specification, you can direct BACKUP to list all of the save sets (including filenames) on the volume:

```
$ BACKUP/LIST MTA0:*
```

If you want to list only the names of the save sets on a volume, without the names of the files in the save sets, you can mount the magnetic tape volume as Files–11; then use the DCL command DIRECTORY. For example:

```
$ MOUNT MTA0: SAVEWORK TAPE
%MOUNT-I-MOUNTED, SAVEWORK mounted on _MTA0:
$ DIRECTORY/SIZE/DATE/PROTECTION TAPE:
Directory MTA0:[]
CONTRACTS.BCK;1          5  13-JUN-1984 12:11  (RWED,RWED,RE,)
JUL20SAVE.BCK;1          3  20-JUL-1984 23:59  (RWED,RWED,RE,)
MYPHILE.BCK;1            2  28-SEP-1984 12:00  (RWED,RWED,RE,)
Total of 3 files, 10 blocks.
```

7.7.14.2 Restoring Disk Files from Save Sets on Magnetic Tape

You can restore all files in the save set to the current default directory as in the following example:

```
$ BACKUP TAPE:NOV12SAVE.BCK []
```

The following command restores files to a subdirectory tree:

```
$ BACKUP TAPE:NOV12SAVE.BCK [LYKINS...]
```

If you want to restore a specific file from a save set, you can use the /SELECT qualifier. For example:

```
$ MOUNT/FOREIGN MTA0:
%MOUNT-I-MOUNTED, NOV25A mounted on _MTA0:
$ BACKUP
_From: MTA0:NOV2SAVE.BCK/SELECT=[LYKINS.GLENDO]STRAT1.DAT;5
_To:   STRAT1.DAT;5
$ DIRECTORY STRAT1.DAT
Directory [LYKINS.GLENDO]

STRAT1.DAT;5

Total of 1 file.
```

In this example, the file STRAT1.DAT in the subdirectory [LYKINS.GLENDO] has been accidentally deleted. The user had previously saved the file in the save set NOV2SAVE.BCK, and now restores it to its original directory by using the /SELECT qualifier.

If you omit a save-set name for an input magnetic tape, BACKUP reads the first save set it encounters on the magnetic tape and stops processing when it reaches the end of that save set. Since BACKUP does not automatically rewind until it reaches the EOT, you can proceed to the next save set (if any exists) by repeating the command. As an alternative, you can include an asterisk (*) with the device specification, which directs BACKUP to read all of the save sets on the magnetic tape volume.

7.7.14.3 Restoring Disk Files from Save Sets on Sequential Disk

You can read a sequential-disk save set either sequentially, or as a normal Files–11 save set.

If you choose to read the save set as sequential, you can mount one volume at a time (as you can do when you create a sequential-disk save set). The default directory for the save-set file specification is the directory [000000], which is the master file directory (MFD) on the disk; therefore, you do not need to include the directory in the input specification. For example:

```
$ MOUNT/FOREIGN DMA0:
$ BACKUP/LOG DMA0:CONTRACTS.BCK/SAVE_SET  CONTRACTS.DAT
%BACKUP-S-CREATED, created DBA2:[USER]CONTRACTS.DAT;1
```

Since the volume is mounted foreign, the DCL command DIRECTORY would fail and generate an error message; however, BACKUP recognizes the format and expects to find the save set CONTRACTS.BCK in the MFD directory. The /SAVE_SET qualifier is required on any operation that restores from disk.

As an alternative, you can read the save set as a normal Files–11 volume:

```
$ MOUNT DMA0: SAVEWORK
$ BACKUP/LOG DMA0:[000000]CONTRACTS.BCK/SAVE_SET  CONTRACTS.DAT
%BACKUP-S-CREATED, created DBA2:[USER]CONTRACTS.DAT;1
```

The default directory is your process default directory, so you must specify the directory [000000]. The SAVE_SET qualifier is necessary for restoring the file to its original structure; without the /SAVE_SET qualifier, the save set would simply be copied.

Save-set files written to disk using the file system can also be read in sequential mode; you may find this mode of operation useful if you use stand-alone BACKUP. You must specify the directory from which the save set is to be read. If the save set was written to a volume set using Files–11 disk structure, you must observe the following rules:

- All volumes of the volume set must be mounted. You must mount the volumes with the /FOREIGN qualifier. (If you are using stand-alone BACKUP, the volumes are implicitly mounted.)

- When you specify the volumes in the input specifier, you must specify the names of the devices in the same order as the relative volume number of the volumes mounted on those devices. For example, the volumes named USER01 and USER02 in a volume set are mounted on two drives, DRA3: and DRA5:. The save-set specifier for the volume set must be in the following format:

```
DRA3:[directory]save-set-name.ext;v, DRA5:
```

7.7.14.4 Restoring Disk Files from Save Sets on Multiple Volumes

If your save set encompasses more than one magnetic tape or sequential disk volume, it is possible to begin restore operations without mounting the first volume of the save set. However, if you use the /IMAGE qualifier, processing must begin with the first volume. If the tape that you mount is not the first volume in the save set, you will receive the warning message:

```
%BACKUP-W-NOT1STVOL, tape 'name' is not the start of a save set
```

If you restore a small number of files from a large save set, it is a good idea to use the /LOG qualifier. This enables you to monitor the files as they are restored. BACKUP will continue processing until it reaches the end of the save set. Once the files you need have been restored, you can press CTRL/Y to terminate processing.

7.8 Managing Disk Space

Parkinson's Law, as applied to public disk storage, states that user files will expand to exceed the available disk storage space. To combat this problem, you can

- Set file expiration dates
- Establish disk quotas

7.8.1 Setting File Expiration Dates

File expiration is a file system feature (available on Files–11 Structure Level 2 disks only) that uses the expiration date of each file to track the file's use. The expiration dates aid the disposal of seldom used files.

To enable the setting of expiration dates, use the DCL command SET VOLUME:

```
$ SET VOLUME device-name: /RETENTION=(min,max)
```

For min and max you specify the minimum and maximum retention periods for files on the volume, expressed as delta time values. For example, the following command sets the minimum retention period to 15 days and the maximum to 20 days:

```
$ SET VOLUME DRA0: /RETENTION=(15-0:0,20-0:0)
```

The retention periods operate as follows: every time a user accesses a file (for either a read or write operation), and that file's expiration date is earlier than the current date plus the minimum retention period, the file's expiration date is updated to the current date plus the maximum retention period. Thus, the expiration date of a frequently accessed file fluctuates between the minimum and maximum period plus the current date. When you set a suitable interval between minimum and maximum retention periods, you can trade between accuracy and efficiency in maintaining expiration dates. The difference between the two periods is the interval at which the expiration date of a frequently accessed file will be updated.

If you specify only a single value in the SET VOLUME /RETENTION command, it becomes the minimum retention period; the maximum retention period is set to twice the minimum or the minimum plus 7 days, whichever is less.

You can simulate the maintenance of "access dates," available in some other operating systems, by setting the retention periods to very small values (for example, 1 hour). Note, however, that doing so will incur substantial overhead in the file system in updating expiration dates so frequently.

This feature does not automatically remove unused files; it maintains expiration dates to permit you to develop your own policy for handling files with little or no activity. For example, the following BACKUP command will copy to tape and then delete all expired files:

```
$ BACKUP/DELETE PUBLIC:[*...]/BEFORE=TODAY/EXPIRED MTA0:ARCH20JUN
```

(Plan to retain the resulting tape for a substantial period of time unless you are unperturbed by user ire.)

Note: **If you start maintaining expiration dates on a previously existing volume, you should be aware that the expiration dates on existing files are zero (until the files are accessed). Files with expiration dates of zero are considered expired.**

7.8.2 Establishing Disk Quotas

You limit the amount of space available to individual users on public volumes (or volume sets) by creating and maintaining quota files on those volumes. Individual users can similarly restrict usage on private volumes. Quotas are maintained and enforced on a per-volume basis. Each volume or volume set has its own quota file; a volume on which quotas are not maintained has no quota file; on a volume set, volume 1 contains the quota file. Each entry in a quota file includes the following information:

- UIC—UIC of a user entitled to maintain files on the volume
- Usage—Number of blocks on the volume taken up by the user's files
- Quota—Maximum number of blocks on the volume that the user's files can take up before an error message is issued
- Overdraft—Number of blocks over the quota that the user's files can take up

The absolute maximum number of blocks permitted a user on a volume is the sum of the quota and the overdraft.

You (or the user maintaining the volume) identify UICs and assign quotas and overdrafts with the Disk Quota Utility (see the *VAX/VMS Utilities Reference Volume*). Usage counts are maintained automatically by the system during normal file activities.

The name of the quota file is [000000]QUOTA.SYS on the applicable volume. A quota file requires one block of disk storage for each 16 entries.

A quota file is initialized with an entry for UIC [0,0]. The usage count for this UIC should not change from 0—the UIC should own no files. Its quota and overdraft, however, serve as defaults in certain situations, and so should be set to values most likely to be assigned as quotas and overdrafts to other UICs.

7.8.2.1 Disk Quota Operations

During normal use of a volume with a quota file, the system automatically updates the usage counts as users create, delete, extend, and truncate files. Users without entries in the quota file are not allowed to create files or allocate space on the volume, unless they have the EXQUOTA privilege.

Exceeding the Quota

To create new files, a user's usage must be below quota (not overdraft). If an operation to add a new file or expand a current file will put a user's usage count over the quota, the system prohibits the operation and issues an error message.

If the rejected operation is an extension of a file opened for write, a user with an overdraft can perform the operation by retrying it. Operations to extend the file will succeed until usage exceeds the sum of the quota and the overdraft. At this point, the system reissues the above message and prohibits further extensions to the file.

Quota restrictions are not enforced for users with the EXQUOTA privilege. However, their usage counts are maintained.

Suspending Quotas

The DISABLE command of the Disk Quota Utility suspends quota operations on a volume; the ENABLE command lifts the suspension. In addition, quota operations on a volume can be suspended at mount time by specifying the /NOQUOTA qualifier to the DCL command MOUNT. Note that when you suspend and then resume quota operations on a volume, you will probably find incorrect usage values in the quota file. You can correct the usage values with the REBUILD command of the

Disk Quota Utility or with the Verify Utility. (The *VAX/VMS Utilities Reference Volume* describes both utilities.)

To discontinue quota operations on a volume, execute the DISABLE command, exit from DISKQUOTA, and delete the QUOTA.SYS file.

Ensuring Quota File Accuracy with REBUILD on Mount

When a volume is mounted that was not properly dismounted the last time it was used, the system performs an automatic REBUILD operation. If quotas are enforced on the volume, this action ensures that the quota file accurately reflects usage of the disk, should any of the following occur:

- The system failed.
- The volume was physically removed before being dismounted.
- The **WRITE PROTECT** button was pushed.

7.8.2.2 Restrictions on Other System Operations

The following restrictions and limitations apply whether or not disk quotas are being used:

- The SYSPRV privilege is required to change the owner UIC of a file, because a change in file ownership consumes the resources of another user.
- Relative volume 1 of the volume set must be online at all times.

8 Installing Images as Known Images

You can enhance the performance of selected executable and shareable images by installing them as known images with the Install Utility (INSTALL). INSTALL is described in the *VAX/VMS Utilities Reference Volume*.

The system defines known images on internal data structures called *known file lists*. Each known file list contains entries for all known images whose device, directory, and file type are identical. For example, all known images with the filename DISK$VOLUME: <MAIN> filename.EXE would be on one known file list, while all known images with the filename DISK$VOLUME: <TEST> filename.EXE would be on another known file list.

Known file lists only last while the system is up. If the system is shut down or fails for any reason, you must reinstall all known images after the system is rebooted. For this reason, the site-independent startup command procedure, SYS$SYSTEM:STARTUP.COM, includes a series of INSTALL commands that install certain system programs as known images. You are encouraged to include in the site-specific startup command procedure, SYS$MANAGER:SYSTARTUP.COM, additional INSTALL commands to install any images that

- Are run frequently
- Are usually run concurrently by several processes
- Require special privileges

(See Chapter 2 for information on the startup command procedures.)

8.1 Assigning Attributes to Known Images

By specifying appropriate INSTALL qualifiers, you can assign known images the following attributes:

- Permanently open—Directory information on the image file remains permanently resident, eliminating the usual directory search required to locate a file. The cost of keeping an image file permanently open is approximately one page of nonpaged dynamic memory per file.
- Header resident—The header of the image file (native images only) remains permanently resident, saving one disk I/O operation per file access. For images with single-block file headers, the cost is less than one page of paged dynamic memory per file; for images with multiblock headers, the cost varies according to the header block count. The images must also be declared permanently open.
- Privileged—Amplified privileges are temporarily assigned to any process running the image (executable images only), permitting the process to exceed its UAF privilege restrictions during execution of the image. In this way, users with normal privileges can run programs that require higher-than-normal privileges.
- Protected—A shareable image contains protected code, that is, code that runs in kernel or executive mode but that can be called by a user-level image. Protected images must be declared shared.
- Shared—More than one user can access the read-only and non-copy-on-reference read/write sections of the image concurrently, so that only one copy of those sections ever need be in physical memory. (Copy-on-reference sections always require a separate copy for each process.) The image must also be declared permanently open.
- Writeable—When a shared non-copy-on-reference writeable section is removed from physical memory (for paging reasons or because no processes are referencing it), it is written back to the image file. Any updates made by processes mapped to the section, therefore, are preserved (while the initial values are lost). The image must also be declared shared.

8.2 Installing Images with the Shared Attribute

Before you install an image with the shared attribute (using the /SHARED qualifier of the Install Utility), you should understand the distinction between *executable* and *shareable* images:

- An *executable* image is one linked with the /EXECUTABLE qualifier (or without the /SHAREABLE qualifier) of the VAX/VMS Linker. (For information on the VAX/VMS Linker, refer to the *VAX/VMS Utilities Reference Volume.*)
- A *shareable* image is one linked with the /SHAREABLE qualifier of the VAX/VMS Linker; it must subsequently be linked into an executable image to be used. (Shareable images are sometimes referred to as linkable images, because they can be specified—implicitly or explicitly—as input files to the link of another file.) A shareable image is not copied into the executable images that link with it. Thus, only one copy of the shareable image need be on disk, no matter how many executable images have linked with it.

When an image is not installed, or is installed without the shared attribute, each process running the image requires private sections in memory. (A shareable image linked to an executable image need not be installed to be executed. At image execution time, the system will create private sections from the shareable image. The only exception is that a shareable image containing a writeable non-copy-on-reference section must be installed as a known image with the shared and writeable attributes.)

When you install an image with the shared attribute, permanent system global sections are created. Execution of non-copy-on-reference global sections requires only one copy per section to be in physical memory, no matter how many processes are running the image to which the sections belong.

The number of images that you can install with the /SHARED qualifier is restricted by the GBLPAGES and GBLSECTIONS system parameters (see the description of the System Generation Utility in the *VAX/VMS Utilities Reference Volume*).

8.3 Installing Images with Privileges

VAX/VMS allows images to execute in an enhanced privilege environment through two mechanisms:

- You can install existing executable images with extra privileges to allow a nonprivileged process to perform the privileged functions of the image.
- You can install privileged shareable images. Privileged shareable images are used to implement user-written system services, thereby allowing nonprivileged images to execute select portions of privileged code, without enhancing the privileges of each image that uses the privileged portion of code.

8.3.1 Privileged Executable Images

You install executable images with enhanced privileges through use of the /PRIVILEGED qualifier of the Install Utility. Only images that are linked with the VAX/VMS Linker qualifiers /NODEBUG and /NOTRACE should be installed with enhanced privilege. Installing other images with enhanced privilege can compromise system security.

8.3.2 Privileged Shareable Images

VAX/VMS supports user-written system services through a mechanism called privileged shareable images. (See the *VAX/VMS Real Time User's Guide* section in the *VAX/VMS Release Notes* for a description of how to create privileged shareable images.) You must link a privileged shareable image with the /PROTECT command qualifier or the /OPTIONS positional qualifier of the VAX/VMS Linker, so that the image acquires its particular form of enhanced privileges:

- Use the /PROTECT command qualifier when all parts of an image require protection.
- Use the /OPTIONS positional qualifier when only part of a privileged shareable image requires protection.

You then install the privileged shareable image with the Install Utility, specifying both the /PROTECTED and /SHAREABLE qualifiers. If you fail to follow all these steps, you will prevent successful activation of a privileged shareable image.

8.4 Deleting Known Images and Dismounting Volumes

System operations are affected by two characteristics of known images:

- Deletion—A known image is not deleted as soon as the /DELETE qualifier is applied. The deletion occurs only after all processes using the image have released it.
- Dismounting—A volume cannot be dismounted while any known file lists associated with it contain entries.

To dismount a volume, then, you must not only delete all known images associated with it, but you must wait for all processes using those images to release them and for the system to write writeable images back to their files. You can use the DCL command SHOW DEVICES/FILES to determine the status of the files.

8.5 Defining Logical Names for Shareable Image Files

At execution time, either a shareable image must reside in the directory SYS$SHARE (which is normally equivalent to SYS$LIBRARY), or you must define a logical name for the file. The logical name must be that used as the input file specification for the shareable image when it was linked. The equivalence string is the current file specification. That is, if a shareable image called STATSHR.EXE does not reside in SYS$SHARE, you must, before running an executable image that calls STATSHR, define the logical name STATSHR. For example:

```
$ DEFINE STATSHR SYS$SYSDEVICE:[TEST]STATSHR
```

The default file type is EXE. If the file specification for STATSHR were SYS$SHARE:STATSHR.EXE, no DEFINE command would be necessary.

Likewise, one shareable image can be substituted for another without requiring the calling executable image to relink. The user simply defines the filename of the old shareable image as the logical name of the file specification of the new one. The following statement defines the filename STATSHR as the logical name of the shareable image SYS$SYSDEVICE:[MAIN]STATSHR.EXE for executable images calling STATSHR:

```
$ DEFINE STATSHR SYS$SYSDEVICE:[MAIN]STATSHR
```

Again the default file type is EXE. (Logical name redirection in the process or group logical name table is ignored when you run a privileged executable image. Only logical names that can be redirected by privileged users are allowed.)

If the new image is installed with the /SHARED qualifier, executable images linked against the old image will be mapped to global sections for the new image. Otherwise, they will be mapped to private sections for the new image.

As demonstrated in the previous examples, the old and new images can have the same name, but must reside in different directories. You should not substitute one version of a file for another in the same directory.

A Note on Installing Shared Images in MA780 Multiport Memory

To install a shared image so that the global sections will reside in a multiport memory unit, you issue the DCL command DEFINE (an ASSIGN command could also be used) to create a logical name of the form:

```
GBL$filename shmem-name:filename
```

The following example ensures that any global sections created for an image whose file name is STATSHR reside in the MA780 multiport memory unit whose logical name is SHRMEM1:

```
$ DEFINE GBL$STATSHR SHRMEM1:STATSHR
$ RUN SYS$SYSTEM:INSTALL
INSTALL> ADD/OPEN/SHARED STATSHR
```

9 Batch and Print Jobs

The batch and print jobs that run on a system each day often comprise a large part of the workload. As system manager, you must become familiar with the number and types of jobs that run on your system and develop ways to submit, schedule, and execute them efficiently. You can manage job workloads and improve system performance by

- Establishing and controlling batch and output queues
- Controlling batch and print jobs
- Establishing and controlling spooled devices

The sections that follow discuss these management tasks.

9.1 Creating and Manipulating Queues

A queue is a list of jobs that are waiting to execute. In VAX/VMS there are three basic types of queues:

- **Execution queue**—A queue that controls batch or print job execution. An execution queue may be a batch or an output queue. Batch queues are described in Section 9.2. Output queues are described in Section 9.3.
- **Generic queue**—A queue assigned to one or more execution queues of the same type. A generic queue holds jobs and places them in an available execution queue. For example, a generic output queue sends jobs to an available printer or terminal queue. Generic output queues are described in Section 9.3.6. Generic batch queues are used on VAXcluster systems and are described in the *Guide to VAXclusters*.
- **Logical queue**—A named queue that is not assigned to a print device when the queue is initialized. Jobs sent to a logical queue cannot execute until you first assign the logical queue to a printer queue, and then start both the printer queue and the logical queue. Logical queues are described in Section 9.3.7.

As system manager, you are responsible for creating and controlling queues. This section introduces the DCL commands used to manage queues and discusses several basic queue management operations. Additional procedures for controlling queues and the jobs they contain are described in Section 9.4.

The commands listed in Table 9–1 allow you to manipulate queues and the jobs that they contain. These commands are described in the *VAX/VMS DCL Dictionary.*

Table 9–1 DCL Commands for Controlling Printer and Batch Queues

Command	Function
ASSIGN/MERGE	Moves all jobs from one queue to another queue
ASSIGN/QUEUE	Assigns an execution queue to a logical queue
DEASSIGN/QUEUE	Deassigns an execution queue from a logical queue
DEFINE/CHARACTERISTIC	Defines a queue characteristic name and number
DEFINE/FORM	Defines a printer form name, number, and attributes
DEFINE/FORM/SETUP	Defines one or more device control library modules that set up the device appropriately for the specified form
DELETE/CHARACTERISTIC	Deletes the definition of a queue characteristic
DELETE/ENTRY	Deletes one or more job entries from a queue
DELETE/FORM	Deletes the definition of a form
DELETE/QUEUE	Deletes a queue
INITIALIZE/QUEUE	Creates a queue
PRINT	Enters a job in a printer or terminal queue
SET DEVICE/SPOOLED	Establishes the spooling of a device
SET QUEUE	Changes the attributes of a queue

Table 9–1 (Cont.) DCL Commands for Controlling Printer and Batch Queues

Command	Function
SET QUEUE/ENTRY	Changes the status or attributes of a job that is not currently executing in a queue
SET RESTART_VALUE	Establishes a test value for restarting portions of batch jobs; used in command procedures
SHOW DEVICES	Displays the status of devices in the system
SHOW QUEUE	Displays information about queues, jobs, forms, and queue characteristics
START/QUEUE	Starts a queue
START/QUEUE/MANAGER	Starts the system job queue manager and opens the queue file (or creates a new queue file if one does not exist); if the /NEW_VERSION qualifier is also specified, forces creation of a new queue file
STOP/QUEUE	Suspends queue operation; executing jobs are suspended, and the queue is placed in a paused state When the following qualifiers are specified, the queue is not paused or stopped: /ABORT — Stops the current print job /ENTRY — Stops the specified batch job /REQUEUE — Stops the specified batch or print job and requeues it to the specified queue
STOP/QUEUE/MANAGER	Stops the system job queue manager and closes the queue file

Table 9–1 (Cont.) DCL Commands for Controlling Printer and Batch Queues

Command	Function
STOP/QUEUE/NEXT	Stops a queue after all executing jobs have completed processing
STOP/QUEUE/RESET	Abruptly stops a queue and aborts all executing jobs
SUBMIT	Enters a job in a batch queue
SYNCHRONIZE	Waits for completion of the specified job

9.1.1 Starting the System Job Queue Manager

The system job queue manager controls the scheduling of batch and print jobs. You start the job queue manager and open the queue file (or optionally create a new queue file) with the DCL command START/QUEUE/MANAGER. A new queue file is created if one does not already exist, or if the /NEW_VERSION qualifier is specified. Use of the START/QUEUE/MANAGER command is restricted to users with both OPER and SYSNAM privileges.

You must include this command before any other queue commands in your site-specific startup command procedure (see Chapter 2). You can specify the name of a queue file as a command parameter. If you omit this parameter, the job controller uses the default file specification SYS$SYSTEM:JBCSYSQUE.DAT.

You can control the initial allocation and extension size of the queue file with the /EXTEND_QUANTITY qualifier. The default value is 100 disk blocks.

9.1.2 Initializing and Starting Queues

To set up a queue, you first create or *initialize* the queue, and then start it. You can use the following commands to initialize and start a queue:

- INITIALIZE/QUEUE/START—Initializes and starts a queue
- INITIALIZE/QUEUE—Initializes a queue
- START/QUEUE—Starts an initialized queue

Use of the INITIALIZE/QUEUE commands is restricted to users with OPER privilege. Use of the START/QUEUE command requires OPER privilege or EXECUTE (E) access to the queue.

With the INITIALIZE/QUEUE/START command, you initialize and start a queue using a single command. This method is recommended in command procedures, specifically the startup command procedure SYS$MANAGER:SYSTARTUP.COM, where it is necessary to initialize and immediately start a queue if it is stopped. For example, the following commands are used to initialize and start the batch queue SYS$BATCH and the printer queues LPA0: and LPB0:

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=2/BASE_PRIORITY=3  SYS$BATCH
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPB0:
```

If the commands are executed when the queue is stopped, the INITIALIZE/QUEUE/START command initializes and starts the queue. Otherwise, if the queue is already running, the command is ignored and no operation occurs. (See Chapter 2 for a description of startup command procedures.)

Setting up queues is not restricted to startup time. In the course of normal operation, you can create queues as needs dictate. Typically, you use the INITIALIZE/QUEUE/START command to create and start a queue. However, separate INITIALIZE /QUEUE and START/QUEUE commands are useful when you want to initialize a queue but start it at a later time. For example, suppose you want to initialize a printer queue and make sure that it is set up correctly before you start the queue. By specifying appropriate qualifiers to the START/QUEUE command, you can update specific queue attributes when you start the queue.

9.1.3 Modifying Queues

Once you initialize a queue, you can change many of its attributes with the DCL command SET QUEUE. The use of this command is restricted to users with OPER privilege or E access to the queue.

The SET QUEUE command allows you to change many queue attributes without having to stop the queue, initialize it, and start it again. For example, the following command modifies the running batch queue, SYS$BATCH:

```
$ SET QUEUE/JOB_LIMIT=4/DISABLE_SWAPPING  SYS$BATCH
```

The command in this example changes the job limit and disables swapping on SYS$BATCH. All other attributes of the queue remain the same. The changed attributes do not affect the execution of current jobs; however, all subsequent jobs are executed with the new attributes in effect.

9.1.4 Suspending Queues

To suspend a running queue, issue the DCL command STOP /QUEUE. This command halts the execution of all current jobs in the queue and places the queue in a paused state. The use of this command is restricted to users with OPER privilege or E access to the queue.

To restart a queue and restart any jobs that are paused, issue the START/QUEUE command.

9.1.5 Stopping Queues

To stop a queue, issue the the DCL command STOP/QUEUE /NEXT. This command allows any currently executing jobs to complete and then stops the queue. Once you issue the command, the queue is stopped, and any new jobs are prevented from executing. If you omit the /NEXT qualifier, the execution of all current jobs is temporarily halted, and the queue is placed in a paused state.

Use of the STOP/QUEUE/NEXT command is restricted to users with OPER privilege or E access to the queue.

To restart a queue, issue the START/QUEUE command.

9.1.6 Deleting Queues

You can delete queues, as necessary, using the DCL command DELETE/QUEUE. You must first stop the queue by issuing the DCL command STOP/QUEUE. When you delete a queue, all jobs in the queue are deleted along with the queue. The use of this command is restricted to users with OPER privilege.

To delete a queue, proceed as follows:

1 Stop the queue by issuing the STOP/QUEUE/NEXT command.

2 Wait for executing jobs to complete.

3 Delete the queue by issuing the DELETE/QUEUE command.

The commands in the following example stop the printer queue ALTFORM, check the contents of the queue, and delete the queue:

```
$ STOP/QUEUE/NEXT  ALTFORM
$ SHOW QUEUE/ALL ALTFORM
$ DELETE/QUEUE  ALTFORM
```

9.1.7 Retaining Jobs in Queues

You can cause a queue to retain processed jobs by specifying the /RETAIN qualifier to the INITIALIZE/QUEUE, START /QUEUE, or SET QUEUE command. The /RETAIN qualifier has the following format:

```
/[NO]RETAIN[=option]
```

You indicate the type of job that the queue is to retain by specifying one of the following options:

Option	Description
ALL	Specifies that jobs be retained in the queue in a completed status after they are executed
ERROR	Specifies that jobs be retained only if they completed unsuccessfully (the completion status has the low bit clear)

By default, jobs are not retained.

The following command specifies that the queue retain all jobs that complete with a "nonsuccess" (low bit clear) status:

```
$ SET QUEUE/RETAIN=ERROR  BATCH_QUE
```

9.1.8 Emptying the Queue File

In the unlikely event that the queue file (SYS$SYSTEM:JBCSYSQUE.DAT) becomes corrupted, it will be necessary to create a new queue file. You should suspect that the queue is corrupted whenever

- You notice irregularities in the display produced by the DCL command SHOW QUEUE.
- Jobs are not being run.
- Jobs seem to disappear.

Before you can create a new queue file, you need to stop all execution queues on the system by issuing the DCL command STOP/QUEUE/MANAGER. This command performs an orderly shutdown of the system job queue manager, stops all execution queues, terminates all executing jobs, and denies any further queue requests. Once all queue processes are terminated, the queue file is closed.

The STOP/QUEUE/MANAGER command is contained in the site-independent shutdown command procedure, SYS$SYSTEM:SHUTDOWN.COM (see Chapter 2).

To create a new version of the queue file, issue the DCL command START/QUEUE/MANAGER/NEW_VERSION.

Use of the STOP/QUEUE/MANAGER and START/QUEUE /MANAGER commands is restricted to users with both OPER and SYSNAM privileges.

9.1.9 Setting Privilege Restrictions on Queues

Operations that apply to a queue or to specific jobs in a queue are controlled by UIC-based protection (see the discussion of protection in the *VAX/VMS DCL Dictionary*) in the same way access to other system objects, such as files, is controlled. The ability to control queue operations through UIC-based protection allows you to restrict the types of jobs and users for a particular queue.

When you initialize a queue, the queue is assigned an owner UIC and a protection mask. Jobs are assigned an owner UIC equal to the UIC of the process that submitted the job. Each operation that is performed on a queue or job in a queue is checked against the owner UIC, protection of the queue and the job, and the privileges of the requester.

All operations are checked as follows:

- **Operations that apply to jobs** are checked against the READ (R) and DELETE (D) protection specified for the queue and the owner UIC of the job. In general, R access to a job allows a user to see the attributes of a job, and D access allows the user to delete the job.
- **Operations that apply to queues** are checked against the WRITE (W) and EXECUTE (E) protection specified for the queue and the owner UIC of the queue. A user with W access to a queue can submit jobs to that queue. Users with E access to a queue may act as the operator for that queue, with the ability to affect any jobs in the queue. Users with operator (OPER) privilege have E access to all queues. OPER privilege also enables users to establish queues and affect accounting.

Table 9–2 summarizes the privileges required for various queue operations.

Table 9–2 Privilege Restrictions for Queue Operations

Required Privilege	Commands
OPER and SYSNAM	START/QUEUE/MANAGER STOP/QUEUE/MANAGER
OPER	DEFINE/CHARACTERISTIC DEFINE/FORM DELETE/CHARACTERISTIC DELETE/FORM DELETE/QUEUE INITIALIZE/QUEUE
OPER, E access to the queue, or R access to the job	SHOW QUEUE SYNCHRONIZE
OPER, E access to the queue, or W access to the queue	PRINT SUBMIT
OPER or E access to the queue	ASSIGN/MERGE ASSIGN/QUEUE DEASSIGN/QUEUE SET QUEUE START/QUEUE STOP/QUEUE STOP/QUEUE/NEXT STOP/QUEUE/RESET
OPER, E access to the queue, or D access to the job	DELETE/ENTRY SET QUEUE/ENTRY STOP /QUEUE/ABORT
No privilege	SET RESTART_VALUE

9.2 Establishing Batch Queues

A batch queue controls the execution of batch jobs. It also defines default system resources for batch jobs that are executed on it. Typically, you must decide on the number of batch queues for your installation and determine such attributes as the following:

- Job limit
- Base priority
- Swap mode
- CPU limits
- Working set quotas and limits

(See Chapter 6 for a discussion of limits, quotas, and priorities.)

When a job executes on a batch queue, the job is assigned the attributes of the queue. Users, however, can explicitly override some queue attributes for a particular job by specifying qualifiers with the DCL command SUBMIT.

Users submit batch jobs to the VAX/VMS system and queue them for execution in two ways:

- As command procedure disk files submitted by issuing the DCL command SUBMIT. These files are placed in a batch queue and selected for execution according to their relative priority in the queue. By default, the name of this batch queue is SYS$BATCH.
- As batch job files submitted by issuing the DCL command JOB from a card reader. These batch job files are spooled onto disk and placed in a batch queue. Unless the JOB command specifies otherwise, the name of this batch queue is also SYS$BATCH.

Before users can submit batch jobs, you must create and start at least one batch queue, as described in the following sections.

In VAX/VMS, more than one batch job can execute at the same time from any given queue. The number of jobs that can execute simultaneously depends on the job limit that you establish for the queue.

After a batch queue has been started, the job with the highest relative priority is the first candidate for execution. For jobs having the same priority, the earliest submitted job is the first candidate for execution.

You should normally encourage users to submit large jobs (such as compiling and linking large programs) as batch jobs, and to reserve interactive use of the system for jobs that do not require extensive resources. Toward this end, you may do the following:

- Restrict the working set size of interactive jobs by providing a small (for example, 300) WSEXTENT value in the UAF records (see Chapter 6).
- Expand the working set size of batch jobs by providing a large (for example, 500) WSEXTENT value when you create the batch queue.

You can likewise restrict and expand time limits on jobs by setting the CPU values.

9.2.1 Setting Up Batch Queues on Predominantly Interactive Systems

The following guidelines are useful in setting up a batch queue for a system that is predominantly interactive:

1. Set up one batch queue named SYS$BATCH, the name of the default batch queue.
2. Give SYS$BATCH the following characteristics:
   - Job limit—1 to 4
   - Base priority—3 or 4
   - Swapping mode—swapping enabled (default)
   - Working set default—300

- Working set quota—200 to 500
- Working set extent—400 or more
- CPU default—infinite (default)
- CPU maximum—infinite (default)

Normally, you would include the following command in your site-specific startup command procedure to create and start this queue (see Chapter 2):

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=2/BASE_PRIORITY=3 -
/WSDEFAULT=300/WSQUOTA=500/WSEXTENT=2000  SYS$BATCH
```

9.2.2 Setting Up Batch Queues on Predominantly Batch Systems

The following rules of thumb are useful in setting up batch queues for a system that is predominantly a batch system and in which editing is the principal interactive activity:

1. Set up three batch queues as follows:
   - SYS$BATCH—the default batch queue with swapping enabled.
   - FAST—a queue for executing high-priority jobs that should not be swapped out of memory.
   - SLOW—a "background" queue for processing low-priority jobs. Typically, these are large jobs with large requirements for physical memory. Usually, it is uneconomical to swap such jobs out of memory. You can adjust the system workload by stopping and restarting background queues as needed.
2. Give SYS$BATCH the following characteristics:
   - Job limit—6 to 10
   - Base priority—4 (by default)
   - Swapping mode—swapping enabled (by default)
3. Give FAST the following characteristics:
   - Job limit—1 (by default)
   - Base priority—4 (by default)

- Swapping mode—swapping disabled

4 Give SLOW the following characteristics:

  - Job limit—1 (by default)
  - Base priority—low (3, for example)
  - Swapping mode—swapping disabled

Normally, you would include the following commands in your site-specific startup command procedure to create and start these queues (see Chapter 2):

```
$ INITIALIZE/QUEUE/START/BATCH/JOB_LIMIT=6  SYS$BATCH
$ INITIALIZE/QUEUE/START/BATCH/DISABLE_SWAPPING  FAST
$ INITIALIZE/QUEUE/START/BATCH/BASE_PRIORITY=3-
/DISABLE_SWAPPING  SLOW
```

9.2.3 Setting Up Batch Queues on Small Systems

On small systems (under 1MB), you should add up the pages required for the batch working sets and ensure that enough fluid memory remains for interactive jobs. Fluid memory can be reassigned from one process to another through swapping and paging. (You can calculate fluid memory as the space that remains when you subtract the number of pages permanently allocated to VAX/VMS from the total memory. To obtain these values, issue the DCL command SHOW MEMORY.)

If fluid memory drops below the value of the WSMAX system parameter, you must reduce the number of batch jobs or make the FAST and SLOW jobs swappable. Otherwise, a deadlock could result.

9.3 Establishing Output Queues

An output queue controls the execution of print jobs. The term *output queue* can mean any one of the following:

- **Printer queue**—An execution queue assigned to a specific line printer device.
- **Terminal queue**—An execution queue assigned to a terminal that is being used as a printer (never interactively). Terminal queues are created and controlled by the same commands as printer queues.

- **Generic queue**—A queue assigned to one or more execution queues. Jobs submitted to a generic queue are transferred to an assigned execution queue when one is available. Section 9.3.6 describes generic queues in more detail.
- **Logical queue**—A queue not assigned to a print device when initialized. Jobs sent to a logical queue cannot execute until you first assign the logical queue to an execution queue, and then start both the execution queue and the logical queue. Section 9.3.7 describes logical queues.

Before users can submit print jobs, you must initialize and start at least one output queue. See Section 9.1.2 for a description of how to initialize a queue.

Print jobs are queued for processing in one of two ways: without the direct intervention of a user (that is, implicitly) or with the direct intervention of a user (that is, explicitly).

Print jobs are queued implicitly when a program or DCL file sends its output to a spooled printer. The output is placed in a file. When the file is closed, it is placed in an output queue. Both the spooling of the output file to an intermediate device and the subsequent queuing of a job consisting of this file occur without the direct intervention of a user. Spooled printers are described in Section 9.3.8.

Print jobs are queued explicitly by use of the DCL command PRINT. Users can explicitly queue a disk file or several files for printing. (The *VAX/VMS DCL Dictionary* describes the PRINT command in detail.) The disk file or files specified in the PRINT command are queued as a print job; if several files make up a print job, they will be printed together.

Unlike batch queues, where several jobs can execute simultaneously, an execution queue can print only one job at a time. Print jobs are scheduled for printing according to their priority; the highest-priority job is the first selected for printing. If several jobs have the same priority, the smallest job is printed first, unless the queue is initialized with the /SCHEDULE=NOSIZE qualifier. Jobs of equal size having the same priority are selected according to their submission time; the job submitted the earliest is printed first.

By default, print jobs queued by use of the PRINT command are placed in the queue named SYS$PRINT. Thus, to use the default version of the PRINT command in a system with only one printer, name the printer queue SYS$PRINT in your site-specific startup command procedure as follows:

```
$ INITIALIZE/QUEUE/START/ON=LPA0:  SYS$PRINT
```

To use the default version of the PRINT command in a system with several line printers of matching characteristics, you would normally establish SYS$PRINT as the name of a generic queue. Generic printer queues are discussed in Section 9.3.6.

9.3.1 Naming a Printer Queue

Normally, when you initialize a printer queue for a particular printer, you assign to the queue the name of the physical device. For example, the following command initializes and starts the queue for the line printer LPA0: and assigns the queue the name LPA0:.

```
$ INITIALIZE/QUEUE/START  LPA0:
```

To create a printer queue and assign it a name that is different from the printer's physical device name, specify the /ON qualifier to the INITIALIZE/QUEUE command:

```
/ON=[node::]device
```

The /ON qualifier allows you to assign a queue name that describes the printer's function. For example, you could assign the name LETTER to the queue for a letter quality printer. Users wanting to print letters could simply queue their jobs to the queue LETTER. The node name is intended for use on VAXcluster systems and is specified by the SYSGEN parameter SCSNODE.

The following command initializes and starts the queue named LETTER, assigning it to the printer LPC0:

```
$ INITIALIZE/QUEUE/START/ON=LPC0:  LETTER
```

9.3.2 Using Device Control Libraries

A device control library is a text library that contains escape sequence modules for programmable printers. You can use device control modules to control various printer functions.

You assign a device control library to the queue for a programmable printer by specifying the /LIBRARY qualifier to the INITIALIZE/QUEUE command as follows:

```
/LIBRARY=filename
```

The filename is the name of the library file containing the desired modules (**do not specify a file type**). By default, the filename is SYS$LIBRARY:SYSDEVCTL.TLB. Libraries must be in SYS$LIBRARY and have the default file type TLB. You can specify the filename only with the /LIBRARY qualifier.

Print operations that request a particular device control module (by way of the PRINT/SETUP or PRINT/FORM command) use the module from the library specified for the queue. If you have a small configuration of printers, and normally use only a few modules, you typically store all modules in a single library and assign that same library to each printer queue. For sites with a large number of printers, you typically create and assign a separate device control library for each printer queue.

Note: **The form name specified with the PRINT command is checked for validity when the command is issued. However, the library module names specified with the /SETUP qualifier are not checked for validity until the file is printed because the name of the device control library is not bound until that time. Therefore, to reduce the chance of error, DIGITAL recommends that users issue the DEFINE/FORM /SETUP command to assign a library module to a specified form. The PRINT/FORM command may then be used to properly set up the device.**

By using separate libraries, you can create modules that have the same name and perform the same functions, but use sequences unique to a specific type of printer. If you use a single library to store modules for different types of printers, you must make sure that each module has a unique name.

For more information on specific escape sequence codes, see the operation guide for the particular printer.

Note: **In order to add or delete a module from a library, any queue currently using the library must be stopped.**

9.3.3 Restricting Printer Queue Processing

To assign printer queue attributes that restrict the types of job that can execute on the queue, specify the following qualifiers to the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command:

Qualifier	Description
/[NO]BLOCK_LIMIT	Restricts job processing according to block size
/CHARACTERISTIC	Restricts job processing according to printer characteristics
/FORM	Restricts job processing according to the form stock mounted on the printer

Using these qualifiers, you can control what types of job can execute on a queue. Sections 9.3.3.1 through 9.3.3.3 describe how to use these qualifiers.

9.3.3.1 Restricting Jobs by Size

In some cases you may want to restrict print job processing on a queue to jobs of a certain size. You can impose limits on the size of jobs that can be printed on a queue by specifying the /BLOCK_LIMIT qualifier to the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. (For a complete description of the qualifier, see the *VAX/VMS DCL Dictionary*.) The /BLOCK_LIMIT qualifier has the following format:

```
/[NO]BLOCK_LIMIT=([lower][,upper])
```

To impose block size limits, you must specify at least one of the following parameters:

Parameter	Description
lower	A decimal number specifying in blocks the smallest job that can be scheduled by the queue. If a print job is submitted that contains fewer blocks than the lower value, the job remains pending until the block limit for the queue is changed. By default there is no lower limit on job size.
upper	A decimal number specifying in blocks the largest file that can be scheduled by the queue. If a print job is submitted that exceeds this value, the job remains pending until the block limit for the queue is changed. By default there is no upper limit on job size.

The /BLOCK_LIMIT qualifier is useful when you want to limit processing on a queue to either small or large jobs. By default, there are no limits on file size for a queue.

9.3.3.2 Restricting Jobs by Characteristics

You can restrict print job processing on a queue by assigning the queue specific printer characteristics. A job can execute on a queue only if each characteristic specified (by name or number with the /CHARACTERISTIC qualifier to the PRINT command) for the job is also specified for the queue. For example, if a job specifies characteristic 25, it can execute only on a queue for which characteristic 25 was specified. The job remains in a "pending" state until a queue with that characteristic becomes available.

Note: **Specification of a characteristic for a queue does not prevent jobs that do not specify that same characteristic from executing.**

You specify the printer characteristics of a printer queue with the /CHARACTERISTIC qualifier to the INITIALIZE /QUEUE, START/QUEUE, or SET QUEUE command. The /CHARACTERISTIC qualifier has the following format:

```
/[NO]CHARACTERISTIC=(characteristic[,...])
```

You can specify a characteristic as a number or a name. You define a characteristic name and a corresponding number with the DCL command DEFINE/CHARACTERISTIC. This command is restricted to users with OPER privilege. Issue the DEFINE/CHARACTERISTIC command as follows:

```
$ DEFINE/CHARACTERISTIC name number
```

The following example defines the characteristic name REDINK as the number 3:

```
$ DEFINE/CHARACTERISTIC REDINK 3
```

Subsequent INITIALIZE/QUEUE, START/QUEUE, SET QUEUE, and PRINT commands can use the number 3 or the name REDINK to describe one printer characteristic.

To delete a defined characteristic, issue the DCL command DELETE/CHARACTERISTIC as follows:

```
$ DELETE/CHARACTERISTIC name number
```

By default, a queue or job has no defined characteristics.

9.3.3.3 Restricting Jobs by Form

You can restrict jobs to a specific printer form by assigning the printer queue a form type. A job can execute on a queue only if the paper stock name of the form specified (by name or number with the /FORM qualifier to the PRINT command) for the job is the same as that specified for the queue. For example, if a job specifies form number 3, it can only execute on a queue if a form on the same stock as form 3 is specified for that queue. The job remains in a "pending" state until a queue with that form becomes available.

You specify the type of form currently mounted in the assigned printer with the /FORM qualifier to the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. The /FORM qualifier has the following format:

```
/FORM=type
```

You can specify a form type as a name or number. You define a form name, a corresponding number, and optional form characteristics with the DCL command DEFINE/FORM. This command is restricted to users with OPER privilege. Issue the DEFINE/FORM command as follows:

```
$ DEFINE/FORM name number [/qualifiers]
```

By specifying additional qualifiers to the DEFINE/FORM command, you can define the following characteristics for a particular form type:

- A specific image area for the form
- A type of paper stock
- Specific device control modules
- Print job processing attributes

For a detailed description of the DEFINE/FORM command and its qualifiers, see the *VAX/VMS DCL Dictionary*.

The command in the following example defines the form name CHECKS as the number 3 and defines the margins for the form:

```
$ DEFINE/FORM CHECKS 3/STOCK=DEFAULT -
_$ /MARGIN=(BOTTOM=3,TOP=3,LEFT=6,RIGHT=6)
```

In this example, once you define a form you can then refer to that form by either name or number. You cannot abbreviate the form name.

The system supplies a form named DEFAULT, number 0, with all default attributes. The default form for both queues and print jobs is number 0. The stock is the VAX/VMS-supplied default.

To delete a defined form, specify the DCL command DELETE /FORM in the format:

```
$ DELETE/FORM name
```

9.3.4 Specifying Print Job Processing Attributes

By specifying qualifiers to the DCL command PRINT, users can control whether file burst pages, file flag pages, or file trailer pages are printed with their jobs. You can control default printing of these pages by assigning appropriate attributes to the queue. You can also control printing of job separation pages as described in Section 9.3.5.

Six pages may be printed: file burst, file flag, file trailer, job burst, job flag, and job trailer. Most sites will use only a subset of the available separation pages at a given time.

Widths greater than 40 characters and less than 200 characters, and lengths of any size greater than 40 characters are supported for file and job separation pages. Pages requested for widths greater than 200 characters are formatted and printed at 200-character widths. Lengths less than 40 characters are extended by that form length until the 40-character threshold is exceeded.

Note: A form size of less than 80 characters by 51 lines may result in less information being displayed.

Page characteristics and information provided are described in Table 9–3.

Table 9–3 Separation Pages

Page	Characteristics and Information
JOB FLAG/BURST	Indicates that a new job follows, which may comprise one or more files. Items are • *Header bar* : single-segment bar composed of the following elements: — Rows of a repeated numeral indicating the sequence of the job in the printer queue — An embedded text string specified by defining the PSM$ANNOUNCE system logical name. The length of the string is limited to one form width. If PSM$ANNOUNCE is not defined, the default text string is "Digital Equipment Corporation" followed by the system version number. ("Digital Equipment Corporation" is abbreviated to "DEC" for shorter form widths.)

Table 9–3 (Cont.) Separation Pages

Page	Characteristics and Information
	• *Note* if specified[1] • *Identification banner* for the process submitting the job: username, job name, and job number • *Job sentence* indicating — Job name and number — Name of queue to which job was submitted — Submission date and time — Process username, UIC, and account — Job priority — Print device name — Job start time — Execution queue name • *Footer bar* similar to *header bar* except that the embedded text string is "Digital Equipment Version V4.0 "
FILE FLAG/BURST	Indicates that a file print follows. Items are • *Header bar* : three-segment bar composed of the following elements: — A central segment identical to the *job header bar* — Flanking segments with rows of a repeated character indicating the sequence of the file in the job • *Note* if specified[1]

[1]Users may specify a string up to 255 characters in length with the /NOTE qualifier of the DCL command PRINT.

Table 9–3 (Cont.) Separation Pages

Page	Characteristics and Information
	• *Identification banner* for the process submitting the job: username, filename • *File sentence* indicating — Full file specification, including device and directory, name, type, version, and revision time and date — File size in blocks — File organization — File owner's UIC — File record characteristics: record type, carriage control, size of longest record • *Job sentence*
FILE TRAILER	Indicates the end of a file print. Items are • *Header bar*: five-segment bar composed of the following elements: — Central segment with "END OF FILE" banner — Inner flanking segments with job-sequence numeral — Outer flanking segments with file-sequence character • *Identification banner* for the process submitting the job: username, filename

Table 9–3 (Cont.) Separation Pages

Page	Characteristics and Information
	• *Qualifier sentence*: indicates the print, queue, and form qualifiers active when the file was printed; nonactive qualifiers (with the exception of /NOFEED) are not presented • *File sentence* • *Job sentence* • *Footer bar* similar to *file flag/file burst footer bar* • *Ruler* : a sequence of numbers counting to the end of the form

Table 9–3 (Cont.) Separation Pages

Page	Characteristics and Information
JOB TRAILER	Indicates the end of a job print. Items are • *Header bar* : three-segment bar composed of the following elements: — Central segment with "END OF JOB" banner — Flanking segments with job-sequence numeral • *Identification banner* of the process that submitted the job; job name and job number • *Receipt box* with the signoff fields **Received:**, **Date:**, and **Operator:** • *Job sentence* • *Footer bar* similar to JOB FLAG /BURST footer bar. • *Ruler*

A *separation burst bar* (the characters VAX/VMS) is printed over the page crease between the job flag and job burst pages, and file flag and file burst pages, to facilitate the separation of job and file printouts.

You assign default print control features to a queue by specifying the /DEFAULT qualifier to the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE command. The /DEFAULT qualifier has the following format:

```
/[NO]DEFAULT=(option[,...])
```

You can specify one or more of the following default options:

Option	Description
[NO]BURST[=value]	Controls whether a file burst page is printed preceding output. If you specify the value ONE, a burst page is printed once with the first file in the job. If you specify the value ALL, a burst page is printed with every file in the job.
[NO]FEED	Controls whether a form feed is automatically inserted at the end of a page.
[NO]FLAG[=value]	Controls whether a file flag page is printed preceding output. If you specify the value ONE, a flag page is printed once with the first file in the job. If you specify the value ALL, a flag page is printed with every file in the job.
[NO]TRAILER[=value]	Controls whether a file trailer page is printed following output. If you specify the value ONE, a trailer page is printed once with the last file in the job. If you specify the value ALL, a trailer page is printed with every file in the job.

If you specify any of these qualifiers without specifying a value, the value ONE is used by default.

By default, no default options are assigned to the queue.

9.3.5 Specifying Job Separation Features

You can control certain functions that the printer performs between jobs it prints by assigning the printer queue job separation features. Unlike default print control features, users cannot use the PRINT command to override the job separation features assigned to a queue.

Job separation features are assigned to a queue by specifying the /SEPARATE qualifier to the INITIALIZE/QUEUE, START /QUEUE, or SET QUEUE command. The /SEPARATE qualifier has the following format:

`/[NO]SEPARATE=(option[,...])`

You assign job separation features by specifying one or more of the following options:

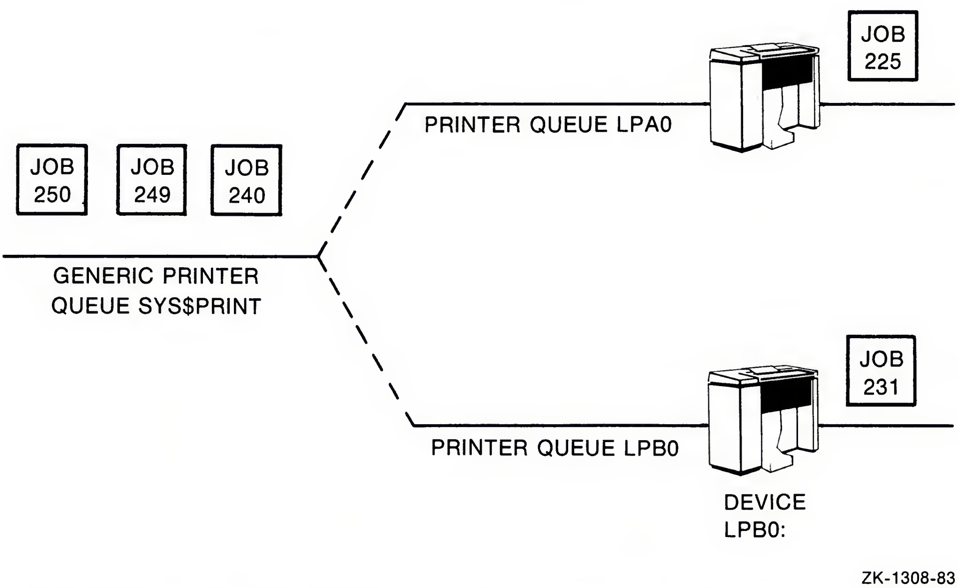
Option	Description
[NO]FLAG	Specifies that a job flag page is printed at the beginning of every job.
[NO]BURST	Specifies that a job burst page is printed at the beginning of every job.
[NO]RESET=(module[,...])	Specifies one or more device control library modules that contain the job reset sequence for the queue. The specified modules from the queue's device control library SYS$LIBRARY:SYSDEVCTL are used to reset the device each time a job reset occurs. The RESET sequence occurs after any file trailer and before any job trailer. Thus, all job separation pages will be printed when the device is in its RESET state. You should assign the RESET job separation feature to queues for printers that support characteristics changeable by the DCL command PRINT/SETUP. The RESET feature ensures that any SETUP features specified for a particular job are not assigned to the next job.
[NO]TRAILER	Specifies that a job trailer page is printed at the end of every job.

By default, no job separation features are assigned to the queue.

9.3.6 Creating Generic Output Queues

When print jobs are sent to a generic output queue, they are held there until one of the assigned printer or terminal queues becomes available. Generic output queues are useful because they distribute the processing of print jobs among similar devices. Figure 9–1 illustrates a generic output queue.

Figure 9–1 Generic Output Queue



If your system has more than one line printer of a similar type, you should create at least one generic output queue. Since print jobs submitted with the PRINT command are sent to the queue SYS$PRINT by default, you should establish SYS$PRINT as a generic queue, assigning to it printers that perform basic print operations.

You establish a generic queue by specifying the /GENERIC qualifier with either the INITIALIZE/QUEUE or START /QUEUE command. See the *VAX/VMS DCL Dictionary* for a description of how the /GENERIC qualifier is used with these commands.

Note: The /DEFAULT, /FORM, and /SEPARATE qualifiers are not meaningful for generic output queues. Consequently, the DCL commands INITIALIZE/QUEUE and START /QUEUE will reject these qualifiers if /GENERIC is also specified. If the attributes of an established generic queue are modified, these three qualifiers may be accepted. However, they will have no effect on the operation of the generic queue.

With the /GENERIC qualifier, you can assign a printer queue to a generic queue by one of the following methods:

- **Default**—Using this method, you do not specify a queue name parameter. Instead, any printer queue (either printer or terminal) that has generic printing enabled is assigned to the generic queue by default.
- **Explicit**—With this method, you specify the names of printer queues (both printer and/or terminal) as parameters.

The next two sections describe how to assign generic queues using each of these methods.

9.3.6.1 Assigning Generic Queues by Default

On systems with a small number of printers, you should use the default function of the /GENERIC qualifier to establish a generic printer queue. With this method, you specify the /GENERIC qualifier without specifying any queue-name parameters, and all printer queues that are enabled for generic printing are assigned to the generic queue.

Printer queues are enabled for generic printing by default when they are initialized. You usually set up queues for line printers that are in remote locations, that use special forms, or that possess unique printer characteristics with generic printing disabled, to exclude them from possible assignment to a generic queue. The qualifier /NOENABLE_GENERIC, when used with the INITIALIZE/QUEUE, START/QUEUE, or SET QUEUE commands, disables generic printing on a printer queue.

The following command procedure creates four different printer queues, including a generic queue:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPAO:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPBO:
$ INITIALIZE/QUEUE/START/NOENABLE_GENERIC  LPCO:
$ INITIALIZE/QUEUE/START/GENERIC  SYS$PRINT
```

In this procedure, since no queue names were specified with the /GENERIC qualifier, only the queues that have generic printing enabled by default, LPA0: and LPB0:, are assigned to the generic queue SYS$PRINT. The queue LPC0: is not assigned to the generic queue because generic printing is explicitly disabled.

There is no advantage in creating more than one generic printer queue by using the default method, since all printer queues with generic printing enabled are assigned to the generic queue. Again, on systems with only a few printers, the default method is adequate.

Using the default method, you can assign to a generic queue only one type of output queue, either printer queues or terminal queues, not both. To establish a generic terminal queue, specify the /TERMINAL qualifier along with the /GENERIC qualifier. If you want to set up a generic queue that serves both printer and terminal queues, you must use the explicit assignment method described in the next section.

9.3.6.2 Assigning Generic Queues Explicitly

On systems with a large number of printers of many different types, you should explicitly assign printer queues to a generic queue using the /GENERIC qualifier. The /GENERIC qualifier accepts a list of the queues. The following command procedure creates four different printer queues, including a generic queue:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPB0:
$ INITIALIZE/QUEUE/START/DEFAULT=(FLAG,TRAILER,NOFEED)  LPC0:
$ INITIALIZE/QUEUE/START/GENERIC=(LPA0:,LPB0:)  SYS$PRINT
```

In this procedure, the printer queues LPA0: and LPB0: are assigned to the generic printer queue SYS$PRINT. The printer assigned to LPC0: uses special forms and is not suited for general printing. Therefore, it is not assigned to the generic queue SYS$PRINT. Print files queued by default to SYS$PRINT are printed on whichever assigned printer (either LPA0: or LPB0:) is free.

By specifying a list of queue names with the /GENERIC qualifier, you can assign printer queues, terminal queues, or both to a generic printer queue. This method also allows you to create more than one generic queue. For example, you may want to create a separate generic queue for each set of printers having similar characteristics:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPB0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPC0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  TTD5
$ INITIALIZE/QUEUE/START/GENERIC=(LPA0:,LPB0:)  SYS$PRINT
$ INITIALIZE/QUEUE/START/GENERIC=(LPC0:,TTD5)  SMALLFORM
```

This command procedure creates two generic queues, SYS$PRINT for routine print jobs, and SMALLFORM for print jobs requiring special size forms. Here it is assumed that printer queues LPC0: and TTD5 are set up for special form jobs.

9.3.7 Creating Logical Queues

A logical queue is a named queue that is not assigned to a printer or terminal when initialized. Jobs sent to a logical queue are held until the logical queue is assigned to a printer queue and both queues are started. Logical queues can be used, for example, to hold print jobs of low-priority users for printing during off-peak hours.

You assign a logical queue to a printer queue with the DCL command ASSIGN/QUEUE. The use of this command is restricted to users with OPER privilege or E access to the queues.

Inserted in a command procedure, the following commands initialize and start the printer queue LPA0: and initialize the logical queue named HOLD:

```
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPA0:
$ INITIALIZE/QUEUE  HOLD
```

Jobs sent to the printer queue LPA0: are queued for printing. However, any jobs sent to the logical queue HOLD are held until HOLD is assigned to LPA0: and started (normally from command level) as follows:

```
$ ASSIGN/QUEUE  LPA0:  HOLD
$ START/QUEUE  HOLD
```

To deassign the logical queue in the example above, use the following procedure:

1 Stop the logical queue by specifying the STOP/QUEUE /NEXT command.

2 Deassign the logical queue by specifying the DCL command DEASSIGN/QUEUE.

The following commands deassign the logical queue HOLD:

```
$ STOP/QUEUE/NEXT  HOLD
$ DEASSIGN/QUEUE  HOLD
```

The *VAX/VMS DCL Dictionary* describes the DEASSIGN /QUEUE command in detail.

9.3.8 Establishing Spooled Devices

Spooling is the technique of using secondary storage to buffer data passing between slow I/O devices (such as line printers and card readers) and physical memory. The slow devices, which can be either the ultimate sources or the ultimate destinations of buffered I/O data, are called spooled devices; the secondary storage devices are called intermediate devices.

As a rule, programs demand input and produce output at irregular intervals during their execution. If programs were allowed to read directly from slow devices and to write directly to slow devices, the execution time of programs would be limited by the speed of the slow devices. If a process were to output data directly to a printer, the process would be tied up for the time it took to print the listing. Also, other processes needing the printer would have to wait.

To balance these input/output demands and enhance throughput, you can establish spooled devices; to all other users, and their programs, the mechanism of spooling is transparent.

Input spooling makes input from a spooled device (such as a card reader) available for processing by placing it in a file on an intermediate device (such as a disk). Input spooling is used to create, from card reader input, batch input files on disk. After batch jobs are spooled to disk, they are queued for processing.

Output spooling makes output from the processor available for transmission to a spooled device (such as a line printer) by placing the output into files on an intermediate device (such as a disk). Output spooling is used to create printer output files on disk. After they are spooled to disk, print jobs are queued for printing.

The actual transfer of inputs from a spooled device to an intermediate device, or of outputs from an intermediate device to a spooled device, is sometimes carried out by processes called symbionts.

Input symbionts read input at the speed of the input spooled device, and buffer it in a file on the intermediate device. Later, when the input is needed, it is read directly from the file on the intermediate device rather than from the spooled device. While one set of input data is being processed, the input symbiont is free to read another set of input data into another file on the intermediate device.

Output symbionts read data from an intermediate device and write the data to an output spooled device at the speed of that output device. The data on the intermediate device is generated by programs that transmit output directly into files on the intermediate device. The I/O waiting time of programs is thus minimized. When an output file is complete, it is queued for printing by an output symbiont. As with input symbionts, two separate jobs can overlap: while an output symbiont is printing a file stored temporarily on an intermediate device, another program can be producing another output file on the intermediate device.

Card readers are spooled by default. To use a card reader without spooling, users must allocate the reader before making it ready to read a card deck. By default, also, the queue SYS$BATCH is used to queue spooled jobs. Thus, no special command is needed to establish card readers as spooled devices.

Most important, on a system with both spooled input devices and spooled output devices, you must create and start at least one batch queue to handle spooled input and one output queue to handle output for each spooled output device.

You enable output spooling for a printer by establishing the printer as a spooled device with the DCL command SET DEVICE/SPOOLED. The use of this command is restricted to users with the OPER privilege. The *VAX/VMS DCL Dictionary* describes the SET DEVICE command in detail.

For each output spooled device, you can use the SET DEVICE command to specify both the printer queue to which spooled files are queued and an intermediate device to which spooled files are written. By default, the queue name is the same as the printer device name and the intermediate device is SYS$DISK.

When you select an intermediate device, ensure that it has sufficient free space for the volume of queued output you expect. If you plan to enforce disk quotas on the intermediate device, ensure that all expected users of the spooled device have a quota authorized on the intermediate device.

The following command enables output spooling for the printer LPA0: and assigns the printer the intermediate device SYS$DISK by default and the generic queue SYS$PRINT:

```
$ SET DEVICE/SPOOLED=SYS$PRINT LPA0:
```

Note, however, that this syntax may lead to errors. Thus, to be safe, you should always specify the intermediate disk and queue explicitly.

Whenever a file is copied to a spooled device, it is queued for printing on the queue specified by the SET DEVICE command when the file is closed. If you assign a generic queue to a spooled device, a job copied to that device is queued to the generic queue, which in turn places the job in one of its assigned queues. As a result, a job copied to LPA0:, for example, may not necessarily print on the printer LPA0:, but instead may print on one of the other devices assigned to the generic queue.

Caution: After establishing a device as spooled, you should test the device, since errors in disk or queue names are not detected until spooling is attempted. You can use a command procedure like the following:

```
!          *****TESTING SPOOLED DEVICE***
$ SET DEVICE/SPOOLED=(SYS$PRINT,SYS$SYSDEVICE:) LPA0:
$ CREATE TEST.LIS
! This is a test of a spooled device.
! This is a test of a spooled device.
^Z
$ COPY TEST.LIS LPA0:
$ EXIT
```

You must decide which devices to include in the system's basic complement of spooled devices. Typically, you set up devices for spooling by making entries in the system startup command procedure.

At a minimum, you should see that at least one line printer is set spooled when the system is booted. In a system with only one line printer, the spooled printer will be the default system printer. Depending on system configuration and anticipated operational needs, more spooled devices can be established at startup. Moreover, in the course of operations (to meet special needs), you can define still other devices as spooled devices without having to reboot the system. Normally, all line printers should be spooled.

You can, as necessary, use the SET DEVICE command to turn off spooling to spooled printers and terminals; however, you must stop the corresponding queues before you can change the spooling status.

9.3.9 Setting Up Printer Queues and Spooled Line Printers

The following rules of thumb are useful in setting up and regulating printer queues, spooled line printers, and terminals used as printers:

- Specify the proper characteristics for each printer with the DCL command SET PRINTER. (The *VAX/VMS DCL Dictionary* describes the SET PRINTER command in detail.)
- Control line overflow with the /TRUNCATE and /WRAP qualifiers of the DCL command DEFINE/FORM (see note below).
- Use the SET DEVICE/SPOOLED command to set each line printer spooled (see Section 9.3.8).
- To produce output on a spooled line printer, initialize a printer queue with the same name as the spooled printer and start that queue. You can also assign a queue a name that is different from the printer's physical device name by using the /ON qualifier with the INITIALIZE command (see Section 9.3.1).

- Assign one printer queue the name SYS$PRINT. For systems with one printer, assign that printer's queue the name SYS$PRINT. If more than one line printer is on the system, make at least one printer queue a generic queue and assign it the name SYS$PRINT. Section 9.3.6 describes how to establish generic printer queues.

A Note on Controlling Line Overflow

For devices controlled by the print symbiont, DIGITAL recommends that you set terminals and printers to avoid wrapping or truncating the print line at the physical limits of the device. In addition, you should use forms definitions to control line overflow, because different forms can have different settings. Users can then switch from form to form without requiring operators to stop the queue, alter the device setup, and restart the queue. Note that forms with the same value for the /STOCK qualifier are automatically mountable without operator assistance.

You control line overflow by using the following qualifiers to the DCL command DEFINE/FORM:

Qualifier	Function
/TRUNCATE	Specifies that data in the right margin be discarded. If the qualifier is not specified, no action is taken.
/WRAP	Specifies that a carriage return and line feed be inserted when the right margin is exceeded. If the qualifier is not specified, no action is taken.

Note that the device driver operates on settings for the /TRUNCATE and /WRAP qualifiers specified with the DCL commands SET PRINTER and SET TERMINAL only *after* the print symbiont has finished formatting data.

Figures 9–2 through 9–4 illustrate some of the most common arrangements of spooled line printers and printer queues. These figures can be used, with the rules of thumb listed above, as guidelines in setting up spooled line printers and printer queues.

9.3.9.1 Single-Printer Configuration

Figure 9–2 illustrates a typical spooling and queuing configuration for a system with one line printer. The following commands in your site-specific SYSTARTUP.COM procedure produce the results listed below:

```
$ SET PRINTER/LOWER  LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG/ON=LPA0: SYS$PRINT
```

1 The characteristics of LPA0: are set.

2 The line printer LPA0: is set spooled.

3 A printer queue for LPA0: is initialized, started, and assigned the name SYS$PRINT.

4 All print jobs are placed by default in a queue named SYS$PRINT and are printed from that queue.

9.3.9.2 Two-Printer Configuration

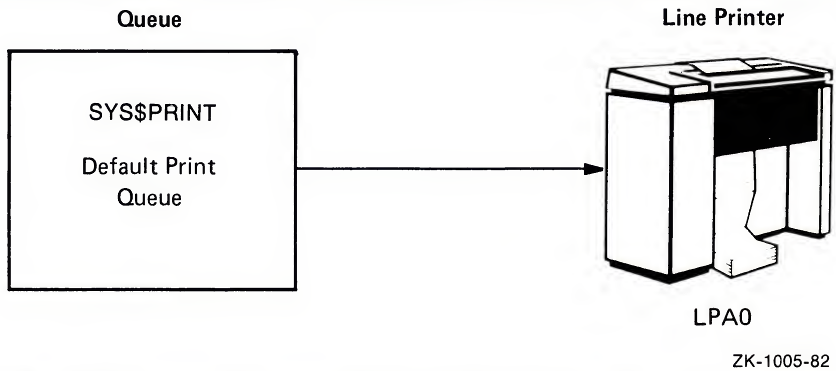
Figure 9–3 illustrates a typical spooling and queuing configuration for a system with two line printers that have the same characteristics. The following commands produce the results listed below:

```
$ !
$ ! Set printer characteristics, set LPA0: spooled,
$ ! and set up queue LPA0:.
$ !
$ SET PRINTER/LOWER LPA0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPA0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPA0:
$ !
$ ! Set printer characteristics, set LPB0: spooled,
$ ! and set up queue LPB0:.
$ !
$ SET PRINTER/LOWER  LPB0:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPB0:
$ INITIALIZE/QUEUE/START/DEFAULT=FLAG  LPB0:
$ !
$ ! Set up the generic queue SYS$PRINT
$ !
$ INITIALIZE/QUEUE/START/GENERIC SYS$PRINT
```

1 The characteristics of LPA0: are set.

2 The line printer LPA0: is set spooled.

Figure 9–2 Setting Up a Printer Queue on a System with One Printer



3 The printer queue LPA0: is initialized and started.

4 The characteristics of LPB0: are set.

5 The line printer LPB0: is set spooled.

6 The printer queue LPB0: is initialized and started.

7 The generic queue SYS$PRINT is initialized and started and assigned the printer queues LPA0: and LPB0: by default (see Section 9.3.6.1).

8 All print jobs explicitly directed with the PRINT command to one of the two printers are placed in the queue associated with the specified printer.

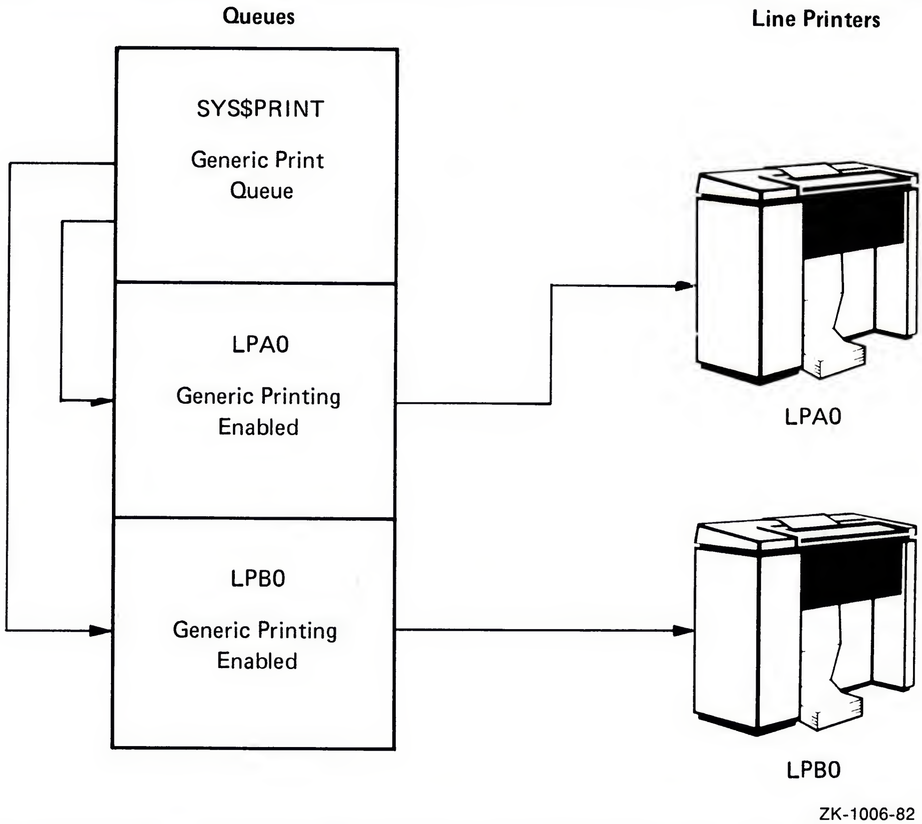
9 Print jobs queued with the PRINT command without device specification are placed by default in the generic queue SYS$PRINT. From the generic queue, jobs are printed on whichever printer is free, by way of either of the two printer queues, LPA0: or LPB0:.

10 Spooled print files destined for either LPA0: or LPB0: are placed in the generic queue SYS$PRINT, which was associated with both these printers. From the generic queue, these jobs are printed on whichever printer is free. (Alternatively, you could set the devices spooled and assign each device a queue name that is the same as the device name, thus ensuring that jobs will print on the same printer to which the file was spooled.)

9.3.9.3 Three-Printer Configuration

Figure 9–4 illustrates a typical spooling and queuing configuration for a system with three line printers: two that have the same characteristics and one that uses special forms, has unique printer characteristics, or is in a remote location. The configuration shown in Figure 9–4 is basically the same as the one in Figure 9–3, with the addition of the spooled printer LPC0: and the creation and starting of the queue LPC0:. Because of the special forms, unique characteristics, or the remote location, printer LPC0: is not suited for general printing. Thus, the following commands produce the same results as those listed with Figure 9–3, with noted exceptions:

Figure 9–3 Setting Up Printer Queues on a System with Two Printers



```
$ !
$ ! Set printer characteristics, set LPAO: spooled,
$ ! and set up queue LPAO:.
$ !
$ SET PRINTER/LOWER  LPAO:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPAO:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START  LPAO:
$ !
$ ! Set printer characteristics, set LPBO: spooled,
$ ! and set up queue LPBO:.
$ !
$ SET PRINTER/LOWER  LPBO:
$ SET DEVICE/SPOOLED=SYS$PRINT  LPBO:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/START  LPBO:
$ !
$ ! Set printer characteristics, set LPCO: spooled,
$ ! and set up queue LPCO:
$ !
$ SET PRINTER/LOWER LPCO:
$ SET DEVICE/SPOOLED=LPCO:  LPCO:
$ INITIALIZE/QUEUE/DEFAULT=FLAG/NOENABLE_GENERIC/START  LPCO:
$ !
$ ! Set up the generic printer queue SYS$PRINT
$ !
$ INITIALIZE/QUEUE/GENERIC/START  SYS$PRINT
```

1 The characteristics for LPC0: are set.

2 The line printer LPC0: is set spooled.

3 The printer queue LPC0: is initialized and started with generic printing disabled.

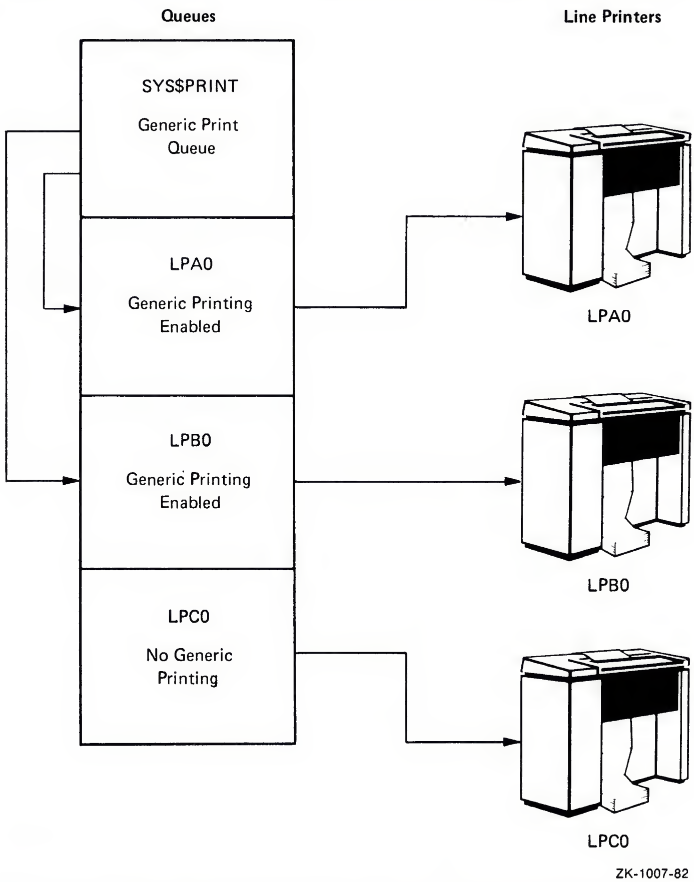
4 Only print jobs explicitly directed to the printer LPC0: are ever placed in the queue LPC0:; no generic printing is ever done on printer LPC0: by way of the queue SYS$PRINT.

9.4 Controlling Printer and Batch Queues

The following sections contain step-by-step procedures for controlling printer and batch queues:

- Merging printer queues
- Resuming print job execution
- Aligning printer forms
- Terminating, requeuing, and holding a print job
- Terminating the execution of a batch job

Figure 9–4 Setting Up Printer Queues on a System with Three Printers



- Removing a batch or print job from a queue
- Restarting the job controller

9.4.1 Merging Printer Queues

When a problem occurs with a print device, you can reroute the queue associated with that device to another device. The following procedure describes how to merge two printer queues without losing jobs in either queue. Note that all commands shown are restricted to users with the OPER privilege or E access to the queue.

1 Stop the queue associated with the malfunctioning print device by issuing the following command:

```
$ STOP/QUEUE/NEXT queue-name1
```

This command inhibits further queuing but permits the current job to be completed, unless the print device is inoperable. If the device is inoperable, use the STOP /QUEUE/RESET command.

2 Requeue the current job by issuing the following command:

```
$ STOP/REQUEUE  queue-name1
```

By requeuing the current job, you ensure that it will be printed from its last checkpoint. Other jobs in the queue will not be queued because the queue is stopped.

3 Take the device offline.

4 Reroute existing jobs queued to the malfunctioning print device to another print device by entering the following command:

```
$ ASSIGN/MERGE queue-name2 queue-name1
```

You should check that the characteristics of the new print device are appropriate for the new jobs.

9.4.2 Resuming Print Job Execution

To restart a halted queue by specifying the position at which the printing of the current job is to resume, enter the START /QUEUE command along with any of the following qualifiers:

Qualifier	Description
/[NO]BACKWARD[=pages]	Specifies the number of pages the file is to be backspaced before printing is resumed. A value of 1 indicates that printing is to resume at the top of the current page. If you omit the page value, the file is backspaced one page.
/[NO]FORWARD[=pages]	Specifies the number of pages the file is to be forward-spaced before printing is resumed. A value of 1 indicates that printing is to resume at the top of the next page. If you omit the page value, the file is forward-spaced one page.
/[NO]SEARCH=string	Specifies that printing is to resume at the page containing the specified string. The search for the string moves forward, beginning on the page following the current page. During the search, consecutive tabs and spaces are treated as a single space, and character case is ignored. Any file positioning specified by /BACKWARD, /FORWARD, or /TOP_OF_FILE is performed before searching.
/TOP_OF_FILE	Specifies that printing is to resume at the beginning of the file.

When multiple positioning qualifiers are used with the same START/QUEUE command, file positioning is performed in the following order:

1. TOP_OF_FILE
2. BACKWARD, FORWARD
3. SEARCH

The command in the following example restarts the queue LPA0: and positions the job to start at page 15.

```
$ START/QUEUE/TOP/FORWARD=15 LPA0:
```

In this example, the file for the job that was being printed when the queue was halted is first positioned at the top of the first page and then positioned 15 pages forward.

9.4.3 Aligning Printer Forms

To print alignment data to aid in aligning printer froms, enter the START/QUEUE command along with the /ALIGN qualifier:

```
/ALIGN=(option1[,option2])
```

The following options control the number of alignment pages and type of alignment data:

Option	Description
MASK	Specifies that input data is masked by replacing alphabetic characters with the character **X** and numerics with the character **9**. Mask characters allow you to prevent the printing of sensitive information. If you omit the MASK option, data is printed unaltered by default.
n	Is a decimal number in the range 1 through 20 that specifies the number of alignment pages to print. By default, one page of alignment data is printed.

You can use the /ALIGN qualifier along with any of the file positioning qualifiers described in the previous section. When /ALIGN is used with any of these qualifiers, file positioning is performed before alignment data is printed. After alignment is complete, the queue is placed in a paused state. You can restart the queue by specifying the START/QUEUE command again. Printing resumes from the point that alignment data started; that is, the task is implicitly backspaced over the pages printed for alignment.

The command in the following example restarts the queue LPA0:.

```
$ START/QUEUE/BACKWARD=2/ALIGN=(MASK,4) LPAO:
```

The file for the job that was being printed when the queue was paused is backspaced two pages before printing resumes. The /ALIGN qualifier prints four pages of alignment mask characters and then pauses the queue.

9.4.4 Terminating, Requeuing, and Holding a Print Job

The following procedures describe how to terminate, requeue, and hold a print job that is currently being printed. You usually perform these tasks only if the owner of a job requests that the job be terminated, or if you observe garbled output on the print device.

To terminate a print job, enter the STOP/ABORT command and specify the name of the queue:

```
$ STOP/ABORT queue-name
```

This command terminates the current job and begins printing the next job in the queue. The use of this command is restricted to users with either OPER privilege, E access to the queue, or D access to the current job.

The following command terminates the printing of the current job on queue LPA0:. The next job in the queue is immediately queued for printing.

```
$ STOP/ABORT   LPAO:
```

To stop and requeue the current job, enter the STOP/QUEUE /REQUEUE command in the following format:

```
$ STOP/QUEUE/REQUEUE=[queue-name] queue-name[:]
```

This command causes an executing print job on the specified queue to be stopped and requeued on the current or on an alternate queue. The queue does not stop, and more jobs are processed if available.

To hold a print job, enter the STOP/QUEUE/REQUEUE /HOLD command in the following format:

```
$ STOP/QUEUE/REQUEUE=[queue-name]/HOLD queue-name[:]
```

When you specify /HOLD, the aborted queue is placed in a hold state for later release with the SET QUEUE/ENTRY /RELEASE or SET QUEUE/ENTRY/NOHOLD commands. If you do not need to process a job that has been relegated to the hold state, you can delete the job with the DELETE/ENTRY command.

Note that if you are requeuing a job on a batch queue, you *must* specify the /ENTRY qualifier; for example:

```
$ STOP/QUEUE/ENTRY=entry-number/REQUEUE=[queue-name] queue-name[:]
```

9.4.5 Terminating the Execution of a Batch Job

The following procedure describes how to terminate the execution of a batch job. You usually perform this task only when careful assessment shows a job must be terminated or the owner of the job requests the termination.

1 Issue the following command to determine the entry number of the batch job and the name of the queue in which the job is located:

```
$ SHOW QUEUE/BATCH/ALL
```

2 Delete the batch job by issuing either one of the following commands:

```
$ STOP/ENTRY=entry-number  queue-name
$ DELETE/ENTRY=entry-number  queue-name
```

Use of the STOP/ENTRY and DELETE/ENTRY commands is restricted to users with either OPER privilege, E access to the queue, or D access to the specified job.

The following example demonstrates how to terminate a batch job.

```
$ SHOW QUEUE/BATCH/ALL
```

```
Batch queue SYS$BATCH
  Jobname          Username      Entry  Status
  -------          --------      -----  ------
  MAILER           DECNET         1701  Holding until  8-JUL-1984 10:33
  ASSEM_JOBCTL     JONES          1152  Holding until 12-JUL-1984 00:00
Batch queue SPECIAL
Batch queue TEXT_PROC
  Jobname          Username      Entry  Status
  -------          --------      -----  ------
  2307SMRCL        MARCO          1719  Executing
  2307SMRDS        MARCO          1720  Pending
```

```
$ STOP/ENTRY=1719  TEXT_PROC
```

Assume a user has observed that job 1719 is a runaway job. The user is not the owner of the job, and lacks sufficient privilege to stop it. The user enlists your aid. You issue the SHOW QUEUE/BATCH/ALL command to determine the queue in which the job has been entered. The display shows that job 1719 is in the TEXT_PROC queue. You then delete the job by issuing the STOP/ENTRY command.

9.4.6 Removing a Batch or Print Job from a Queue

The following procedure describes how to delete a batch or print job that has been entered in a queue but has not yet been processed. You usually perform this task only after careful assessment that it is necessary or at the request of the owner of the job.

1. Enter the following command to determine the job's entry number and the name of the queue in which the job is located:

   ```
   $ SHOW QUEUE/DEVICE/ALL
   ```

2. Delete the job by issuing the following command:

   ```
   $ DELETE/ENTRY=entry-number queue-name
   ```

Use of the DELETE/ENTRY command is restricted to users with either OPER privilege, E access to the queue, or D access to the specified job.

The following example demonstrates how to remove a print job from a queue.

```
$ SHOW QUEUE/DEVICE/ALL

Printer queue NORMAL, on LPA0:
  Jobname         Username      Entry  Blocks  Status
  -------         --------      -----  ------  ------
  TUBING          DOOLEY        1724      891  Printing
  TRICKS          VERNOSH       1725      962  Pending
Printer queue LETTER, on LPB0:
Generic printer queue SYS$PRINT

$ DELETE/ENTRY=1725 NORMAL
```

A user has requested that job 1725 be deleted. You issue the SHOW QUEUE/DEVICE/ALL command to determine in which queue the job has been entered. The display shows that job 1725 is in the NORMAL queue. You then abort the job by issuing the DELETE/ENTRY command and by specifying the entry number 1725.

9.4.7 Restarting the Job Controller

The job controller is designed to restart after a failure; however, if you must restart the JOB_CONTROL process manually, first stop all symbiont processes that have a UIC of [1,4] and a name of the following form:

```
process name SYMBIONT_nnnn
```

Then invoke the STARTUP.COM procedure as follows:

```
$ @SYS$SYSTEM:STARTUP JOBCTL
```

9.5 Managing the Card Reader

The card reader is used to read card decks. Users may submit the two following types of card decks for processing:

- Batch job card decks
- Data card decks

To ensure that card decks are processed efficiently, you must understand their characteristics and the use of the card reader. The following sections describe which cards you should check before processing a deck through a card reader and how to determine which cards are damaged. Section 9.5.3.2 outlines a procedure for processing a card deck through the card reader.

9.5.1 Distinguishing Types of Card Deck

Before loading a card deck into the card reader, you should determine:

- Whether the deck is a batch job or a data deck, because their processing requirements differ
- Whether the card reader is set to the correct translation mode

The following sections describe how to make these determinations.

9.5.1.1 Batch Job Card Deck

A batch job card deck consists of three segments:

- The initial cards
- The program cards
- The last card

The initial two cards in a batch job card deck are the $JOB and the $PASSWORD cards. These cards log in the user and the batch job to the system. Following the initial two cards are program cards. Program cards contain instructions that direct the system to libraries, routines, and data needed to complete the batch job. The last card must be either an end-of-job command ($EOJ) card or an end-of-file (EOF) card. Either of these cards tells the system that this is the end of the job.

Note: **For more information on card reader batch jobs from a system user's viewpoint, refer to the *Guide to Using DCL and Command Procedures on VAX/VMS.***

Checking Input

The system cannot execute the job without $JOB and $PASSWORD cards. If you are given a card deck with these cards omitted, you should return the deck so that the user can insert them.

Since the card deck contains the user's password, you must ensure that it is always handled with care to preserve the security of the user's account.

The last card in the deck must be either an $EOJ or an EOF card.

If the last card is not one of these end cards, you can type one on the card punch (12-11-0-1-6-7-8-9 overpunch in column 1) and insert it at the end of the deck.

Checking Output

The log file produced by a card reader batch job is queued for printing to the default system printer queue, SYS$PRINT. To have the log file queued to a different queue, the user can specify the /PRINTER qualifier on the $JOB card.

If an error occurs while the system is attempting to validate the $JOB and $PASSWORD cards, OPCOM sends to the card operator an error message that reports the job card and the error.

9.5.1.2 Data Card Deck

A data deck contains data that will be either read by a program or copied to a file for later use. The process that will read the data deck is usually associated with an interactive user at a terminal, or else with a batch job that is submitted by an interactive user. Since the user and process already are logged in to the system, the first card can contain any data the user wants to specify. Then, either the program must read the exact number of cards supplied, or the last card should be an EOF card to inform the program that this is the end of the data deck.

When a user wants a data deck to be read, you should ensure that the user has allocated the card reader. If the card reader is not allocated, the system tries to submit the deck as a batch job and subsequently just flushes the deck through the reader, rejecting the job.

If the program does not read the exact number of cards (as with the COPY command), the EOF card must be the last card in the deck, to inform the program that this is the end of the deck. Without this card, the program waits indefinitely for more cards, and the system prints "card reader offline" messages on the operator's terminal. If the card deck lacks an EOF card, you can type one on a card punch and insert it at the end of the deck.

9.5.2 Setting Card Reader Translation Modes

For the system to read input properly, the card reader must be set to the correct translation mode—the same as the translation mode of the card punch used to prepare the deck. VAX/VMS supports 026 and 029 card punches. (Translation modes are discussed in detail in the *VAX/VMS I/O Reference Volume* and the *Guide to Using DCL and Command Procedures on VAX/VMS*.)

Therefore, these conditions must exist for you to be able to set the card reader to the correct translation mode:

- The first card in the deck must be the translation mode card.
- You must know the mode in which the cards were punched.

To set the translation mode of the card reader for many decks of the same type, use the SET CARD_READER command. This command is fully described in the *VAX/VMS DCL Dictionary*. By default, when the system is bootstrapped, the translation mode is set to 029.

9.5.3 Tending the Card Reader

The job of tending the card reader includes:

- Replacing physically defective cards
- Operating the card reader

9.5.3.1 Replacing Physically Defective Cards

Even when the card deck contains all the required cards, the card reader may not be able to read the deck. This problem is usually caused by one or more physically defective cards.

If the deck contains a faulty card, one of the error indicators located on the front panel of the card reader lights up when the card is read. The card reader goes offline, and operator intervention is required to put it back online. Table 9–4 lists the error indicators, the reasons why they may light up, and the operator action required to correct the situation.

9.5.3.2 Operating the Card Reader

The following steps describe how to load and process a card deck through a card reader.

1. Remove the card weight from the input hopper. Place the cards, face down and with column 1 on the left, in the hopper. Ensure that the first card to be read is at the bottom of the hopper.

   Do not pack the input hopper so full that the air from the blower cannot riffle the cards. If the cards are packed too tightly, the vacuum picker cannot operate properly.

2. Press the **RESET** button. The **HOPPER CHECK** error indicator and the **STOP** light will go out and the cards will be read.

   If the card deck is too large to fit in the input hopper, you can load the excess cards while the reader is operating if you maintain tension on the front portion of the deck.

3. Remove the cards from the output stacker when either the **HOPPER CHECK** error indicator or the **STOP** light is lit.

   If you accidentally press the **STOP** button while the card deck is being read, return the last card in the output hopper to the bottom of the input hopper and press the **RESET** button.

4. If the cards are not read properly after the **RESET** button has been pressed, refer to Table 9–4 for recovery procedures.

Table 9–4 Card Reader Errors: Causes and Corrective Actions

Error	Causes	Corrective Action
READ CHECK	Card edges torn Punch in column 0 or 81	Remove the faulty card from the output stacker, duplicate the card, place it in the input hopper, and press the **RESET** button. If READ CHECK occurs for all cards, the read logic of the card reader is malfunctioning.
PICK CHECK	Damage to leading edge Torn webs Cards stapled together	Remove the card from the input hopper, duplicate the faulty card, place the card back in the input hopper, and press the the **RESET** button. If there is no evidence of card damage, check for excessive warping of the card deck and /or a buildup of ink glaze on the picker face.
STACK CHECK	Jam in the card track Badly mutilated card	Correct the jam and/or remove the mutilated card from the output stacker, duplicate the card, place it in the input hopper, and press the **RESET** button.
HOPPER CHECK	Input hopper empty Output stacker full	Load the input hopper. Unload the output stacker.

9.5.4 Running the Input Symbiont Interactively

You can run the input symbiont interactively, taking card image input from a VAX RMS file, by issuing the following commands:

```
$ DEFINE/USER SYS$INPUT filename
$ RUN SYS$SYSTEM:INPSMB
```

Running the input symbiont interactively requires the following:

- CMKRNL privilege
- Read access to the UAF
- Write access to the default directory of every user

All messages are issued to the terminal instead of the card operator.

10 Errors and Other System Events

The system provides several tools for recording and reporting errors and other system events. These tools include facilities for

- Logging and reporting system events
- Logging operator messages
- Handling user requests
- Reporting problems to DIGITAL

In the event of a severe system failure, VAX/VMS automatically shuts down the system and produces a crash dump of the state of the system at the time the error was detected. You can analyze the dump to help you determine the cause of the system failure. (See the description of the System Dump Analyzer in the *VAX/VMS Utilities Reference Volume*.) In a few cases, as described in Section 10.4, you may want to forward the dump to DIGITAL with a Software Performance Report.

10.1 Error Logging Operations

The system automatically writes error messages to the latest version of a file named SYS$ERRORLOG:ERRLOG.SYS. You can display the information in this file by issuing the DCL command ANALYZE/ERROR_LOG. This command is described in the *VAX/VMS DCL Dictionary*.

The Error Logging Facility consists of three parts:

1. A set of executive routines that detect errors and events and write relevant information into error log buffers in memory
2. A process called ERRFMT, which periodically empties the error log buffers, transforms the descriptions of the errors into standard formats, and stores the formatted information in a file on the system disk

3 The Analyze/Error_Log Utility, activated by the DCL command ANALYZE/ERROR_LOG, which you use to report selectively the contents of an error log file (see the *VAX/VMS Utilities Reference Volume*)

The executive routines and the ERRFMT process operate continuously without user intervention. The routines fill the error log buffers in memory with raw data on every detected error and event. When one of the available buffers becomes full, or when a time allotment expires, ERRFMT automatically writes the buffers to ERRLOG.SYS.

Sometimes a burst of errors can cause the buffer to fill up before ERRFMT can empty them. You can detect this condition by noting a skip in the error sequence number of the records reported in the error log reports. As soon as ERRFMT frees the buffer space, the executive routines resume preserving error information in the buffers.

The ERRFMT process displays an error message on the system console and deletes itself if it encounters excessive errors while writing the error log file. You can restart ERRFMT by invoking the startup command procedure in the [SYSEXE] directory and by specifying the parameter ERRFMT as follows:

```
$ @SYS$SYSTEM:STARTUP ERRFMT
```

Invoke the command procedure from the system manager's account.

10.1.1 Using Error Reports

The error reports generated by the Analyze/Error_Log Utility are useful in two basic ways:

1. They aid preventive maintenance by identifying areas within the system that show potential for failure.

2. They aid the diagnosis of a failure by documenting the errors and events that led up to it.

The detailed contents of the reports are most meaningful to DIGITAL field service personnel. However, you can use the reports as an important indicator of the system's reliability. For example, using the DCL command SHOW ERROR, you may see that a particular device is producing a relatively high

number of errors. You can then use the Analyze/Error_Log Utility to obtain a more detailed report and decide whether you should consult DIGITAL Field Service. In that case, field service personnel can run diagnostic programs to investigate the device and attempt to isolate the source of the errors.

In the event a system component does fail, a Field Service representative can study the error reports of the system activity leading up to and including the failure. For example, if a device fails, you can generate error reports immediately after the failure. One report might describe in detail all errors associated with the device that occurred within the last 24 hours; another report might summarize all types of errors for all devices that occurred within the same time period. The summary report can put the device errors into a system-wide context. The Field Service representative can then run the appropriate diagnostic program for a thorough analysis of the failed device. Using the combined error logging and diagnostic information, the Field Service representative can proceed to correct the device.

Error reports allow you to anticipate potential failures. In turn, Field Service personnel rely on the reports as an aid to both preventive and corrective maintenance. Overall, effective use of the Error Log Utility in conjunction with diagnostic programs, can significantly reduce the amount of system downtime.

10.1.2 Maintaining the Error Log Files

Since the error log file, SYS$ERRORLOG:ERRLOG.SYS, is a shared file, ERRFMT can write new error log entries while other entries in the same file are being read and reported by the Error Log Utility.

ERRLOG.SYS will increase in size and remain on the system disk until it is explicitly renamed or deleted. Therefore, you must devise a plan for regular maintenance of the error log file.

One method is to rename ERRLOG.SYS on a daily basis. This action causes a new error log file to be created and allows the old file (which was renamed) to be copied to a backup volume where it can be kept as long as needed. For example, you could rename the current copy of ERRLOG.SYS to ERRLOG.OLD every morning at 9:00 o'clock. To free space on the system disk, you could then back up the renamed version of the error log file on a different volume and delete the file from the

system disk. Note that you should exercise caution to ensure that error log files are not deleted inadvertently. You may also want to adopt a naming convention for your files that incorporates in the filename a beginning or ending date for the data.

10.1.3 Printing the Error Log Files

The following steps describe how to generate an error log report for all entries in the error log file and how to print the report:

1. Ensure that you have the SYSPRV privilege. You need this privilege to access the error log file.

2. Set your default disk and directory to SYS$ERRORLOG.

3. Examine the error log directory to see which error log file you wish to analyze.

4. To obtain a full report of the current error log file, issue the command:

   ```
   $ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS
   ```

5. Print a copy of the report, using the filename specified with the /OUTPUT qualifier:

   ```
   $ PRINT ERRORS.LIS
   ```

Example

```
$ SET PROCESS/PRIV=SYSPRV
$ SET DEFAULT SYS$ERRORLOG
$ DIRECTORY
Directory SYS$SYSROOT:[SYSERR]
ERRLOG.OLD;2   ERRLOG.OLD;1   ERRLOG.SYS;1
Total of 3 files.
$ ANALYZE/ERROR_LOG/OUTPUT=ERRORS.LIS ERRLOG.OLD
$ PRINT ERRORS.LIS
```

10.2 Using the Operator Log File

The operator log file (SYS$MANAGER:OPERATOR.LOG) is a system management tool you can use to

- Anticipate and prevent hardware and software failures
- Monitor user requests for disk and magnetic tape operations

The log file records system events and user requests sent to the operator terminal by the operator communication process (OPCOM), even when all operator terminals have been disabled. The following message types appear in the operator's log file:

- Initialization of the operator's log file
- Status reports for devices attached to the system
- Terminals enabled and disabled
- Volume mounts and dismounts
- User requests and operator replies
- Changes to system parameters through the System Generation Utility
- Security alarm messages
- DECnet–VAX status messages

By regularly examining the operator log file, you can often detect tendencies toward problems and can thereby ensure that preventive action is taken.

Example 10–1 illustrates some typical messages found in the operator log file.

Example 10–1 Sample Operator Log File (SYS$MANAGER:OPERATOR.LOG)

```
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:33:54.07  %%%%%%%%%%%
Operator '_node-name$terminal-name:' has been disabled, user JONES
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:34:15.47  %%%%%%%%%%%
Operator '_node-name$terminal-name:' has been enabled, user SMITH
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:34:15.57  %%%%%%%%%%%
operator status for '_node-name$terminal-name:'
PRINTER, TAPES, DISKS, DEVICES
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:38:53.21  %%%%%%%%%%%
request 1, from user PUBLIC
Please mount volume KLATU in device MTA0:
Gort, the tape is in cabinet A
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:39:54.37  %%%%%%%%%%%
request 1 was satisfied.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:40:23.54  %%%%%%%%%%%
message from user SYSTEM
Volume "KLATU       " mounted, on physical device MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:40:38.02  %%%%%%%%%%%
request 2, from user PUBLIC
MOUNT new relative volume 2 () on MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:41:07.54  %%%%%%%%%%%
message from user SYSTEM
Volume "KLATU       " dismounted, on physical device MTA0:
15-APR-1984 22:42:14.81, request 2 completed by operator OPA0
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:42:15.83  %%%%%%%%%%%
request 3, from user PUBLIC
MOUNT new relative volume 3 () on MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:42:44.54  %%%%%%%%%%%
message from user SYSTEM
Volume "BERADA      " mounted, on physical device MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:42:44.73
message from user SYSTEM
Volume "BERADA      " dismounted, on physical device MTA0:
I'm sorry, but we are out of tapes.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:45:11.45  %%%%%%%%%%%
request 3 aborted by operator OPA0
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:45:40.54  %%%%%%%%%%%
message from user SYSTEM
Volume "BERADA            " dismounted, on physical device MTA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:46:47.96  %%%%%%%%%%%
request 4, from user PUBLIC
```

(Continued on next page)

Example 10–1 (Cont.) Sample Operator Log File (SYS$MANAGER:OPERATOR.LOG)

```
_TTB5:, This is a sample user request w/ reply expected.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:47:38.50  %%%%%%%%%%%
request 4 was canceled
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:48:21.15  %%%%%%%%%%%
message from user PUBLIC
_TTB5:, This is a sample user request w/o a reply expected.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:49:07.90  %%%%%%%%%%%
Device DMA0: is offline.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:49:20.22  %%%%%%%%%%%
Mount verification completed for device DMA0:
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:49:37.64  %%%%%%%%%%%
Device DMA0: has been write locked.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:53:54.52  %%%%%%%%%%%
Device DMA1: is offline.
Mount verification in progress.
%%%%%%%%%%%  OPCOM, 15-APR-1984 22:54:16.56  %%%%%%%%%%%
Mount verification aborted for device DMA1:
%%%%%%%%%%%  OPCOM, 15-APR-1984 23:33:54.07  %%%%%%%%%%%
message from user NETACP
DECnet shutting down
```

10.2.1 Maintaining the Operator Log File

The operator log file normally resides on the system disk in the [SYSMGR] directory. Because this file is in ASCII format, you can print it. You should print copies regularly and retain these copies for reference. The next section describes how to print copies of the operator log file.

A new version of OPERATOR.LOG is created each time the system is rebooted. (You can also use the DCL command REPLY/LOG to create a new version of the file.) The highest version is always the one in use and is inaccessible.

You should devise a plan for regular maintenance of these files. One way is to rename the second-highest version on a daily basis. The procedure for renaming the operator log file is the same as that described in Section 10.1.2 for renaming the error log file. Do not delete versions that have not been backed up.

10.2.2 Printing the Operator Log File

The following steps illustrate how you can produce a printed copy of the most recent version of the operator log file.

1 Close the current log file and open a new one by entering the following command:

```
$ REPLY/LOG
```

2 Set the default to SYS$MANAGER and issue the following command to list all versions of the file:

```
$ DIRECTORY OPERATOR.LOG
```

3 Rename the second-highest version to OPERATOR.OLD:

```
$ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
```

The version number, -1, specifies that the second-highest version of this file is to be renamed. The highest version number is the current operator log file.

4 Obtain a printed copy of the operator log file by issuing the following command:

```
$ PRINT OPERATOR.OLD
```

Example

```
$ REPLY/LOG
%%%%%%%%%%%  OPCOM, 15-APR-1984 12:29:24.52  %%%%%%%%%%%
logfile initialized by operator _MARS$VTA2:
logfile is SYS$MANAGER:OPERATOR.LOG
$ SET DEFAULT SYS$MANAGER
$ DIRECTORY OPERATOR.LOG
Directory SYS$SYSROOT:[SYSMGR]
OPERATOR.LOG;582          OPERATOR.LOG;581
Total of 2 files.
$ RENAME OPERATOR.LOG;-1 OPERATOR.OLD
$ PRINT OPERATOR.OLD
```

10.2.3 Restarting OPCOM

You can restart OPCOM if for some reason it is deleted or suspended. Simply invoke the STARTUP command procedure in the [SYSEXE] directory and specify one parameter, as in the following example:

```
$ @SYS$SYSTEM:STARTUP OPCOM
```

Invoke the STARTUP command procedure from the system manager's account.

10.3 Using the Operator Terminal to Handle User Requests

One of the chief operator functions, communicating with users in handling requests, must be performed at terminals that have been defined as operator terminals. You can set up an operator terminal by issuing the DCL command REPLY/ENABLE at the desired terminal. When you enter this command, an OPCOM message in the following format will be displayed at the terminal and written to the operator's log file:

```
%%%%%%%%%%%  OPCOM, dd-mmm-yyyy hh:mm:ss.cc, %%%%%%%%%%%
Operator '_nodename$terminal-name:' has been enabled, username 'USERNAME'
```

This message tells you which terminal has been established as an operator terminal and when it was established.

Request Message Formats

To communicate with you, users can issue the DCL command REQUEST, specifying either the /REPLY or /TO qualifier.

If the user issues a REQUEST/REPLY command, the request is displayed at the operator terminal in the following format:

```
%%%%%%%%%%%  OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
request 'request-id' from user 'USERNAME' '__terminal-name:', "message-text"
```

This message tells you which user sent the request, the time it was sent, the request identification number, the originating terminal, and the request itself.

If the user issues a REQUEST/TO command, the request is displayed at the operator terminal in a format similar to the one above, but without a request identification number. The message will have the following format:

```
%%%%%%%%%%% OPCOM, dd-mmm-yyyy hh:mm:ss.cc %%%%%%%%%%%
request from user 'USERNAME' '__terminal-name:', "message-text"
```

When a user issues a REQUEST/REPLY command and you have disabled all operator terminals with the appropriate REPLY/DISABLE commands, OPCOM returns a message to the user indicating that no operator coverage is available. However, all subsequent user requests are recorded in the operator log file.

To respond to user requests, issue the REPLY command with one of the following qualifiers:

- /ABORT=identification-number "message-text"
- /PENDING=identification-number "message-text"
- /TO=identification-number "message-text"

The /ABORT qualifier indicates that the user request has been canceled. The /PENDING qualifier places the request in a waiting state until it can be fulfilled or aborted. The /TO qualifier indicates that the request has been fulfilled.

For more information on the REPLY command, including all the command qualifiers, see the *VAX/VMS DCL Dictionary*.

10.4 Reporting Software Problems

To inform DIGITAL about problems with the VAX/VMS operating system or about errors in VAX/VMS software documents, you should use the Software Performance Report (SPR), which is reproduced as Figure 10–1. Directions for completing the SPR form accompany the form itself.

A supply of SPR forms is included in the VAX/VMS Software Distribution Kit; you can obtain more forms from an SPR center. The addresses of these centers are listed on the backs of the forms.

This section offers advice on how to use this service most effectively, by describing the information that should be provided with all SPRs. Depending on the problem, this information will vary in quantity and content. Remember that the more information you include, the easier it will be for DIGITAL to resolve the problem.

Figure 10–1 Software Performance Report (SPR)

digital

SOFTWARE PERFORMANCE REPORT

FIELD NO.:	CORPORATE SPR NO.:

279086

PAGE ______ OF______

TO SET UP FOR PROPER ALIGNMENT, START AT MARK BELOW.

OPERATING SYSTEM	VERSION	SYSTEM PROGRAM OR DOCUMENT TITLE	VERSION OR DOCUMENT PART NO.	DATE
[				

NAME:
FIRM:
ADDRESS:
CUST. NO.:

DEC OFFICE

DO YOU HAVE SOURCES? YES ☐ NO ☐

REPORT TYPE/PRIORITY

☐ PROBLEM/ERROR
☐ SUGGESTED ENHANCEMENT
☐ OTHER

1. ☐ HEAVY SYSTEM IMPACT
2. ☐ MODERATE SYSTEM IMPACT
3. ☐ MINOR SYSTEM IMPACT
4. ☐ NO SIGNIFICANT IMPACT
5. ☐ DOCUMENTATION/SUGGESTION

SUBMITTED BY: PHONE:

CAN THE PROBLEM BE REPRODUCED AT WILL? YES ☐ NO ☐

ATTACHMENTS

MAG TAPE ☐ FLOPPY DISKS ☐ LISTING ☐ DECTAPE ☐

OTHER:

COULD THIS SPR HAVE BEEN PREVENTED BY BETTER OR MORE DOCUMENTATION? PLEASE EXPLAIN IN PROVIDED SPACE BELOW. YES ☐ NO ☐

CPU TYPE	SERIAL NO.	MEMORY SIZE	DISTRIBUTION MEDIUM	SYSTEM DEVICE	DO NOT PUBLISH ☐

ALL SUBMISSIONS BECOME THE PROPERTY OF DIGITAL EQUIPMENT CORPORATION.

SHORT NAME	MNT. CAT.	MNT. GRP.	XFER GRP.	PL	PRB. TYPE
DATE RECEIVED (MAIL)		DATE TO MAINTAINER	XFER DATE	LOGGED ON	
DATE RECEIVED (ASG)		DATE RECEIVED FROM MAINTAINER	DATE ANSWERED	LOGGED OFF	

EN 1044H-07-R479 (35C)

ADMINISTRATIVE SERVICES GROUP, SWS

ZK-794-82

10.4.1 The Problem Environment

You should supply a complete description, usually in the form of a batch log or console listing, that shows exactly how the problem evolved. **Merely supplying erroneous output or paraphrases of the session is not enough**. You must provide a complete scenario depicting what you did. The problem may be caused by an interaction between various system events, software packages, devices, SYSGEN parameters, DCL symbols, or logical names. Consider including some or all of the displays these commands produce:

```
$ SHOW LOGICAL/ALL
$ SHOW SYMBOL/ALL/GLOBAL
$ RUN SYS$SYTEM:SYSGEN
$ SYSGEN>  USE ACTIVE
$ SYSGEN>  SHOW /ALL
$ SYSGEN>  SHOW /SPECIAL
```

10.4.2 The Problem Scope

As much as possible, eliminate all extraneous elements from the scenario you provide. For example, if the execution of a very complex program causes a problem, strip the program to just the code involved or write a small program that reproduces the problem. This action may help you locate logic errors, and will enable DIGITAL to isolate the difficulty more quickly.

10.4.3 Machine-Readable Files

Supply any software needed to reproduce the problem, if possible. This may include source programs, image files, sample data, command procedures, and other items. When you submit source programs, be sure to include any libraries or require files that you reference. These files should all be provided in machine-readable format. It is best to include console or magnetic tape media with an SPR.

If a system failure has occurred, then include the system dump file. Write the data onto a Files–11 Structure Level 2 disk or an ANSI magnetic tape. The following commands will copy the system dump file to an ANSI magnetic tape:

```
$ MOUNT/FOREIGN MTAO: DUMPS
$ BACKUP SYS$SYSTEM:SYSDUMP.DMP/IGNORE=NOBACKUP  MTAO:DUMP/SAVE_SET
```

You can copy files to the console medium after invoking SYSGEN and issuing the following command:

```
SYSGEN> CONNECT CONSOLE
```

Now remove the console medium and place a scratch medium in the console device.

```
$ INITIALIZE CSA1: SPRDATA
$ MOUNT CSA1: SPRDATA
$ CREATE/DIRECTORY CSA1:[SPR]
$ BACKUP MYDATA.DAT,MYIMAGE.EXE/IGNORE=NOBACKUP CSA1:[SPR] DATA/SAVE_SET
$ DISMOUNT CSA1:
```

When you provide machine-readable data, always include the exact instructions used to write the medium and instructions for reading the medium. (Avoid using other media formats, since doing so can cause unnecessary problems. For example, using FLX without /IM creates an unusable dump file since FLX eliminates all bytes of zero.)

Note: All machine-readable media you submit with SPRs will be returned to you.

10.4.4 System Environment

Every site runs a different type of workload. Some problems appear only under certain conditions. For example, some sites give different classes of users different base priorities. These sites may encounter problems that other sites do not. Such background information can be extremely important in resolving the problem, especially for system hangs or system failures.

You should describe any special software packages that are running, and mention any unusual hardware devices or user-written drivers on your system.

Also include the various version numbers of different pieces of software. For example, if a problem occurs while accessing local symbols during a DEBUG session, specify the version numbers of DEBUG and all compilers or assemblers used.

Sometimes DIGITAL forwards special patches or prereleases of patches for problems that are seriously affecting the system. If you are using any such patches, mention them in the SPR.

10.4.5 User Analysis (Optional)

Optionally you can include an analysis of what you believe is causing the problem. Include miscellaneous information, such as "the problem could only be reproduced when xyz happened," or "the problem does not occur on Version x.y."

DIGITAL requires different kinds of information to resolve different kinds of problems. Use Table 10–1 as an initial checklist to begin collecting the proper information to forward with your SPR.

Table 10–1 Typical Problem-Specific Information for SPRs

Problem	Information to Collect
Command language interpreters	A listing obtained from the DCL commands SHOW SYMBOL/ALL /GLOBAL and SHOW LOGICAL/ALL.
Compiler or assembler	A machine-readable copy of the source program that caused the problem, including all require files and libraries that are referenced. (Try to limit the scope of the problem.)
	A description of the compiler (or assembler) and operating system, including the version of each.
Devices	A copy of the error log at the time of the problem.

Table 10–1 (Cont.) Typical Problem-Specific Information for SPRs

Problem	Information to Collect
Executive	A listing of the active values of the system parameters. (Invoke SYSGEN and specify the USE ACTIVE, SHOW /ALL, and SHOW /SPECIAL commmands.)
	A machine-readable copy of the source program showing the problem.
	A copy of the error log at the time of the problem.
Files	A listing of all the information available on the file, obtained with the DCL command DIRECTORY/FULL; or a dump with the header of the file's directory, obtained with the DCL command DUMP /HEADER.
	A machine-readable copy of the file itself.
Intermittent	A copy of the error log at the time of the problem.
Job controller	A copy of the console printout and a machine-readable copy of the file SYS$SYSTEM: SNAPSHOT.DAT, which is produced when the job controller aborts.

Table 10–1 (Cont.) Typical Problem-Specific Information for SPRs

Problem	Information to Collect
Librarian	A machine-readable copy of the library itself. Machine-readable copies of all input files to the library. A listing of all the information available on the library file, obtained from the DCL command DIRECTORY/FULL. A listing of the library contents, obtained from the DCL command LIBRARY/LIST/FULL. (If the problem cannot be reproduced, describe the scenario and include any command files used.)
Linker	Machine-readable copies of the object files and libraries used in the link. A full map obtained by the DCL command LINK /MAP/FULL.
Machine check	A copy of the error log at the time of the error. A machine-readable copy of the system dump file.

Table 10–1 (Cont.) Typical Problem-Specific Information for SPRs

Problem	Information to Collect
Network	A description of the configurations of the systems involved in the problem, including the versions of the operating systems and DECnet–VAX, the hardware on both systems, and the patch level of the DECnet software on the non-VAX/VMS system, if applicable.
System bugcheck/Failure	A machine-readable copy of the system dump file. (Do *not* send output from the System Dump Analyzer, since it usually does not provide enough information to resolve the problem. If you send the dump file, DIGITAL can always run SDA to obtain as much information and possibly more.) A copy of the error log at the time of the error to help DIGITAL determine if the system problem was triggered by hardware errors.

Table 10–1 (Cont.) Typical Problem-Specific Information for SPRs

Problem	Information to Collect
System hang	A copy of the system dump file, obtained after you crash the system in the manner described in the software installation card for your VAX processor. This causes the system to bugcheck in a manner that is recognizable as a forced crash. Include the console listing and a description of the currently running workload.
Terminals	A list of terminal characteristics produced by the DCL command SHOW TERMINAL. A description of the type of terminal, the type of modem (if any), any special front-end equipment, and any unusual terminal configuration.

11 System Generation

When your system is installed or upgraded, a DIGITAL-supplied command procedure, AUTOGEN.COM in SYS$UPDATE, automatically sets the values of system parameters, the sizes of the primary paging, swapping, and dump files, and the contents of the default installed image list (VMSIMAGES.DAT in SYS$MANAGER). AUTOGEN makes these adjustments after evaluating your hardware configuration and estimating typical workloads.

In many cases, AUTOGEN optimizes your new system when it is invoked during routine installation or upgrade procedures. You may therefore wish to monitor performance for a period of time before making further adjustments.

If (based on system performance and suggestions in the *Guide to VAX/VMS Performance Management*) you decide to modify some system parameters or change the size of a system file, you should perform such modifications with AUTOGEN rather than with SYSGEN, because AUTOGEN analyzes your modifications and adjusts any related parameters. Sections 11.1 through 11.5 describe AUTOGEN functions and usage.

There are, however, certain system modifications that you must perform using SYSGEN. These are described in Sections 11.6 through 11.12.

11.1 AUTOGEN Functions

AUTOGEN performs one or more of the following operations on an upgraded or newly installed system:

- Collects two types of data:

  1. Hardware configuration characteristics.

2 Current values of SYSGEN parameters. Four sets of values (listed in Tables 11–5 through 11–8 at the end of this chapter) are collected in the files OLDSITE1.DAT through OLDSITE4.DAT in SYS$SYSTEM and conditionally written to SYS$SYSTEM:PARAMS.DAT. (For detailed descriptions of SYSGEN parameters, refer to the *VAX/VMS Utilities Reference Volume*.)

- Calculates appropriate new values for significant SYSGEN parameters (listed in Table 11–9) and sizes for the system page, swap, and dump files
- Resets system parameter values and file sizes if necessary
- Optionally shuts down and reboots the system

You should carefully examine the PARAMS.DAT file that AUTOGEN creates at your site during routine installation or upgrade procedures to determine whether

- AUTOGEN has drawn the correct conclusions about your hardware configuration.
- The system parameter values shown are appropriate for your workload requirements.

If you are satisfied with the data in PARAMS.DAT, further adjustments are probably unnecessary.

If, on the other hand, you need to correct configuration data or modify system parameters to accommodate special conditions at your site, invoke AUTOGEN as described in the next section and follow the procedures outlined in Sections 11.3 and 11.4.

11.2 Invoking AUTOGEN

You can invoke AUTOGEN any time after an installation or an upgrade to reset parameter values and system file sizes. The new values and file sizes take effect the next time the system is bootstrapped.

To ensure that you have the required privileges, invoke AUTOGEN from the system manager's account. The format is:

```
$ @SYS$UPDATE:AUTOGEN [start-phase] [end-phase] [execution-type]
```

You can enter up to three parameters to designate the AUTOGEN operation you desire. Note that all parameters are optional; however, missing leading parameters must be replaced by null arguments (that is, " ").

Table 11–1 lists the phase parameters and their effects, including the files needed as input and the files created or changed for output. Table 11–2 summarizes execution types.

All files except VMSIMAGES.DAT reside in the directory specified by the SYS$SYSTEM logical name. VMSIMAGES.DAT resides in SYS$MANAGER.

The start phase must either precede or be identical to the end phase according to the sequence shown in Table 11–1 (the end phase defaults to the same value as the start phase). GENPARAMS is the default start phase; GENFILES may not be specified as the start phase.

Table 11–1 AUTOGEN Phase Parameters

Phase	Input Files	Output Files	Function
SAVPARAMS	None	OLDSITE*.DAT	Save significant old parameters for propogation and update.
GETDATA	OLDSITE*.DAT MODPARAMS.DAT	PARAMS.DAT	Collect all data that will be required by the GENPARAMS and GENFILES phases, including configuration data, old parameters, and site-specific items.
GENPARAMS	PARAMS.DAT	SETPARAMS.DAT VMSIMAGES.DAT	Generate new system parameters; create the installed image list.
TESTFILES	PARAMS.DAT	SYS$OUTPUT	Display the system page, swap, and dump file sizes calculated by AUTOGEN. (See Section 11.4 if you wish to specify sizes.)

Table 11–1 (Cont.) AUTOGEN Phase Parameters

Phase	Input Files	Output Files	Function
GENFILES	PARAMS.DAT	PAGEFILE.SYS SWAPFILE.SYS SYSDUMP.DMP	Generate new system page, swap, and dump files if appropriate. Cannot be specified as the start phase.
SETPARAMS	SETPARAMS.DAT	AUTOGEN.PAR	Run SYSGEN to set system parameters specified by the SETPARAMS.DAT and to generate a new AUTOGEN.PAR file.
SHUTDOWN	None	None	Prepare the system to await a manual reboot.
REBOOT	None	None	Automatically reboot the system.

Table 11–2 Execution Types

Type	Meaning
INITIAL	Specifies that AUTOGEN is being executed as part of an initial system installation. The SAVPARAMS phase is never executed in this case.
V4UPGRADE	Specifies that AUTOGEN is being executed as part of an upgrade from a Version 4.n system or that interactive tuning is being performed. V4UPGRADE is the default execution type.
V3UPGRADE	Specifies that AUTOGEN is being executed as part of an upgrade from a Version 3.n system to a Version 4.0 system.

The commands and examples in the next sections illustrate AUTOGEN procedures you would use to adjust a typical Version 4.n system after an upgrade. You can adapt the procedures, with minor changes, for initial Version 4.0 installations and upgrades from Version 3.n to Version 4.0 systems.

11.3 Collecting Data for AUTOGEN Calculations

To collect the data AUTOGEN needs to perform its calculations, issue the following command:

```
$ @SYS$UPDATE:AUTOGEN SAVPARAMS GETDATA
```

This command instructs AUTOGEN to

- Create OLDSITE*.DAT files that store system parameter data from the current system
- Collect system version, identification, and hardware configuration data
- Collect user-specified data, if any, from MODPARAMS.DAT (see Section 11.4)
- Write data to a PARAMS.DAT file for use in subsequent calculations or propagation to the new system

Example 11–1 shows a typical PARAMS.DAT file. The first group of items is a list of AUTOGEN's conclusions about the system version, system ID, and hardware components. (Table 11–3 defines these items.) You should scrutinize the Boolean items to verify that AUTOGEN has drawn the correct conclusions about your hardware configuration.

The next items are system parameter values collected in OLDSITE*.DAT files or specified in MODPARAMS.DAT. Parameters preceded by an OLD_ prefix will be used within calculations. Other parameters will be written to SETPARAMS.DAT and propagated to the new system.

Note: **Values displayed in PARAMS.DAT as input from MODPARAMS.DAT override corresponding configuration values and values from OLDSITE*.DAT files.**

Example 11–1 Sample AUTOGEN PARAMS.DAT File

```
VERSION="Version 4.0.    "
CPUTYPE=1
SID=20447489
MEMSIZE=16384
DISKSPEED=4
SMALLDISK="false"
NUM_DISK=21
NUM_TAPE=2
NUM_SCOM=9
NUM_CARD=0
NUM_TERM=81
NUM_LP=2
NUM_REALTIME=0
NUM_BUS=1
NUM_MAILBOX=0
NUM_JOURNAL=9
NUM_MISC=0
NUM_CI=1
DRIVER_NPAGEDYN=165385
DECNET="true"
CLUSTER="true"
JOURNALING="false"
! Parameters specified in SYS$SYSTEM:OLDSITE1.DAT
OLD_GBLPAGES=5616
OLD_GBLSECTIONS=150
OLD_GBLPAGFIL=1024
OLD_MAXBUF=2048
OLD_MAXPROCESSCNT=110
OLD_VIRTUALPAGECNT=20480
! Parameters specified in SYS$SYSTEM:OLDSITE2.DAT
CLISYMTBL=80
LRPSIZE=576
! Parameters specified in SYS$SYSTEM:OLDSITE3.DAT
DEADLOCK_WAIT=1
PAGFILCNT=5
REALTIME_SPTS=40
SAVEDUMP=1
! Parameters specified in SYS$SYSTEM:MODPARAMS.DAT
ADD_MAXPROCESSCNT=10
ADD_NPAGEDYN=10000
CLUSTER="false"
SRPCOUNT=1200
SHUTDOWN_TIME=10
```

Table 11–3 Configuration Parameters Specified in Example 11–1

Parameter	Specifies
VERSION	Version number of VAX/VMS used to create this file
CPUTYPE	Code for the VAX processor model number as specified by the $GETSYI system service
SID	System identification number
MEMSIZE	Physical memory size, in pages
DISKSPEED	Code for speed of system disk; values are 1, 2, or 4, where 4 is the fastest category of disks
SMALLDISK	System disk size category ("true": 53,000 blocks or less; "false": more than 53,000 blocks.)
NUM_DISK	Total number of disk devices
NUM_TAPE	Total number of magnetic tape devices
NUM_SCOM	Total number of system communication devices
NUM_CARD	Total number of card readers
NUM_TERM	Total number of terminal lines available
NUM_LP	Total number of printers
NUM_REALTIME	Total number of real-time devices
NUM_BUS	Total number of busses
NUM_MAILBOX	Total number of mailboxes
NUM_MISC	Total number of miscellaneous devices
NUM_JOURNAL	Reserved for future use
NUM_CI	Total number of computer interconnects (CIs)
DRIVER_NPAGEDYN	Bytes of nonpaged pool taken up by drivers
DECNET	Whether DECnet is installed
CLUSTER	Whether node is member of a VAXcluster
JOURNALING	Reserved for future use

11.4 Modifying AUTOGEN Calculations

If, after examining the PARAMS.DAT file, you decide you wish to correct hardware configuration data, modify system parameter values, or explicitly specify sizes for the system page, swap, or dump files (that is, override the sizes calculated by AUTOGEN), follow the steps outlined in this section.

Depending on the adjustments you are making, you may want to invoke AUTOGEN several times, specifying different parameters.

1 Edit the file SYS$SYSTEM:MODPARAMS.DAT as follows:

   - To specify new parameter values, use DCL assignment statements of the form:

     ```
     parameter = parameter-value   ! comment
     ```

     You can also use an ADD_ prefix with the SYSGEN parameters listed in Table 11–9 to increment their values in subsequent AUTOGEN calculations during the GENPARAMS phase. For example:

     ```
     ADD_MAXPROCESSCNT=10
     ADD_NPAGEDYN=10000
     ```

   - To specify system file sizes explicitly, use the keywords PAGEFILE, SWAPFILE, and DUMPFILE followed by an equal sign and the size of the file in blocks. Specifying a value of zero for any of these keywords instructs AUTOGEN not to modify the size of the corresponding file. During the GENFILES phase, AUTOGEN will set the file sizes you specify.

   The parameter values and file sizes specified in MODPARAMS.DAT will be copied into PARAMS.DAT during the next GETDATA phase, and AUTOGEN will make appropriate adjustments in its calculations.

2 Verify the modifications you have made in MODPARAMS.DAT. Issue the command

   ```
   $ @SYS$UPDATE:AUTOGEN GETDATA
   ```

   and examine the resulting PARAMS.DAT file (see Example 11–1).

If you wish to make further modifications to the data in PARAMS.DAT, edit MODPARAMS.DAT and repeat the GETDATA phase.

Caution: **Under no circumstances should you attempt to edit PARAMS.DAT. All modifications to this file must be made by editing MODPARAMS.DAT and (re)executing AUTOGEN's GETDATA phase.**

3 When you are satisfied with the data in PARAMS.DAT, issue the following command:

```
$ @SYS$UPDATE:AUTOGEN GENPARAMS SETPARAMS
```

AUTOGEN will generate the known image file list, VMSIMAGES.DAT, and it will create SETPARAMS.DAT, using in its calculations the parameter values you specified in MODPARAMS.DAT. It will then run SYSGEN to set values for your upgraded system. The new values are propagated to SYS$SYSTEM:AUTOGEN.PAR along with values AUTOGEN either derives in its calculations or supplies by default.

4 Finally, when you are ready to reboot the system, issue the command:

```
$ @SYS$UPDATE:AUTOGEN REBOOT
```

Note that you can include the AUTOGEN parameter SHUTDOWN_TIME in MODPARAMS.DAT to specify the number of minutes before shutdown occurs in this operation.

Once you complete the AUTOGEN REBOOT operation, you can verify the changes in AUTOGEN.PAR. Invoke SYSGEN and display the parameters with the SYSGEN command SHOW /ALL.

11.5 Manipulating System Files

As described in the previous section, you can use AUTOGEN to adjust the sizes of the primary paging, swapping, and dump files. However, you cannot use AUTOGEN to perform any operations on secondary paging, swapping, or dump files. To create, install, or modify secondary files, you must use SYSGEN (see the *VAX/VMS Utilities Reference Volume*).

Note: **If you have moved the swapping file away from SYS$SYSTEM, specify a value of 0 for the keyword SWAPFILE in MODPARAMS.DAT. This instructs AUTOGEN not to create a new swapping file.**

You can move most of the paging file away from the system disk. However, you must always maintain at least part of the paging file, PAGEFILE.SYS, on the system disk, SYS$SYSTEM. In fact, since the paging file on the system disk must contain enough paging space for system processes and VAX RMS global buffers, you should retain at least several thousand blocks for PAGEFILE.SYS.

When there are multiple paging files on a system, the system allocates page file space for a new process from the paging file that has the most available space. Thus, you can control some of the I/O activity on the disks with paging files by adjusting the size of the paging files. To achieve nearly balanced activity, make all the paging files of equal size. To direct activity away from the system disk, make the required paging file on SYS$SYSTEM a minimal size while you make one or more additional paging files much larger. The extremely large files will be chosen to provide paging space for each new process because the large files should be the ones with the most available space.

By following the procedures in the previous section, you can use AUTOGEN to create a paging, swapping, or dump file that is smaller than the current version of the file. After you have booted and begun using the new file, remember to use the DCL command DELETE or PURGE to reclaim the disk space from the old version of the file.

11.6 SYSGEN Functions

Using SYSGEN, you can manipulate the following parts of the VAX/VMS operating system:

- System parameters—Create and modify a standard system parameter file for use in subsequent conversational bootstrap operations; modify the parameter values in the current system parameter file for use in subsequent bootstrap operations; dynamically modify the parameter values of the active system (applies only to the dynamic system parameters)
- Devices and device drivers—Connect devices and load their device drivers (most of this work is automatic)
- System files—Create additional paging and swapping files
- Startup command procedure—Identify the current site-independent startup command procedure
- Multiport (shared) memory—Initialize multiport memory units
- MSCP server—Load and start the MSCP server (see the *Guide to VAXclusters*).

See the *VAX/VMS Utilities Reference Volume* for full details of the System Generation Utility and its commands.

11.7 Modifying System Parameters

The bootstrap process initializes the active system parameter values in memory from the current system parameter file on disk (that is, the starting parameter values are those in SYS$SYSTEM:VAXVMSSYS.PAR). In a conversational bootstrap operation, you can modify these values by reinitializing the active parameter values from a parameter file or the default list, and by setting new parameter values on an individual basis. At the end of the bootstrap operation, the system parameter file is modified to conform to the active parameter values.

Warning: **Many of the system generation parameters can affect other parameters or the performance of the system. It is suggested that you make changes to system parameters by editing**

the file SYS$SYSTEM:MODPARAMS.DAT and reinvoking AUTOGEN.

After the system is booted and running, you can run SYSGEN to create or modify parameter files, including the current system parameter file, and to modify the dynamic parameter values of the active system. Follow these steps:

1. Invoke SYSGEN—Invoking SYSGEN initializes a *work area* for the active parameter values.

2. Optionally issue a USE command—You can reinitialize the work area to include the values of a new parameter file, the current system parameter file, or the default values, if the active values do not provide a suitable base for subsequent operations.

3. Issue SET commands—You modify parameters on an individual basis. These modifications have no effect outside the SYSGEN work area.

4. Issue a WRITE command—You create a parameter file, modify the current system parameter file on disk, or modify the active system in memory.

During these operations, you can use the SHOW command to examine the parameter values in the SYSGEN work area.

11.7.1 Creating a New Parameter File

The creation of a new parameter file does not immediately affect the system. During a subsequent conversational bootstrap operation, however, you can initialize the active system with the values of the new file. The following example creates a new version of the AUTOGEN.PAR system parameter file with a new value for the REALTIME_SPTS parameter:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYSGEN
SYSGEN>  USE AUTOGEN
SYSGEN>  SET REALTIME_SPTS 10
SYSGEN>  WRITE AUTOGEN
SYSGEN>  EXIT
```

The next example creates a user file named SYS$SYSTEM:OURSITE.PAR, using the AUTOGEN.PAR file as a base:

```
  SET DEFAULT SYS$SYSTEM
  RUN SYSGEN
SYSGEN>  USE AUTOGEN
SYSGEN>  SET REALTIME_SPTS 10
SYSGEN>  WRITE OURSITE
SYSGEN>  EXIT
```

11.7.2 Modifying the System Parameter File

Modification of the current system parameter file also does not immediately affect the system. During subsequent bootstrap operations, however, the active system is initialized with the new values. A conversational bootstrap operation permits you to modify these values further, while a nonstop bootstrap operation makes the new values the values of the active system. The following example modifies the REALTIME_SPTS parameter value in the system parameter file:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  USE CURRENT
SYSGEN>  SET REALTIME_SPTS 10
SYSGEN>  WRITE CURRENT
%OPCOM, 15-APR-1984 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process
ID 00160030 into file VAXVMSSYS.PAR
SYSGEN>  EXIT
```

11.7.3 Modifying the Active System

Modification of the active system immediately affects that subset of the system parameters called the dynamic parameters by changing their values in memory. The discussion of the System Generation Utility in the *VAX/VMS Utilities Reference Volume* identifies the dynamic parameters (as does the SYSGEN command SHOW/DYNAMIC). The other parameters regulate structures that cannot be changed while the system is running. The following example illustrates how you modify the active value of the PFCDEFAULT parameter:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  SET PFCDEFAULT 127
SYSGEN>  WRITE ACTIVE
%OPCOM, 15-APR-1984 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITEACT, ACTIVE system parameters modified by process
ID 00160030 into file VAXVMSSYS.PAR
SYSGEN>  EXIT
```

Modification of the active system does not affect the current system parameter file on disk. If, for example, you set new active parameter values (WRITE ACTIVE) and later want to use them for subsequent bootstrap operations, the values must be explicitly written to the current system parameter file on disk:

```
$ RUN SYS$SYSTEM:SYSGEN
SYSGEN>  WRITE CURRENT
%OPCOM, 15-APR-1984 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process
ID 00160030 into file VAXVMSSYS.PAR
SYSGEN>  EXIT
```

11.8 Connecting Devices and Loading Device Drivers

Normally, you issue the AUTOCONFIGURE command to connect automatically all devices physically attached to the system and to load their device drivers, saving a great deal of effort and reducing the possibility of error. Devices not attached to the system and devices with nonstandard names can be connected and their device drivers loaded with explicit CONNECT (or CONNECT and LOAD) commands. You must exercise great care in issuing CONNECT and LOAD commands. (See the discussion of the System Generation Utility in the *VAX/VMS Utilities Reference Volume* and the *Guide to Writing a Device Driver for VAX/VMS*.)

Devices not connected automatically by AUTOCONFIGURE include the network communications logical device and the console block storage device. To connect the network communications logical device, use the following explicit CONNECT command:

```
SYSGEN> CONNECT NET/NOADAPTER/DRIVER=NETDRIVER
```

To connect the console block storage device, use the following explicit CONNECT command:

```
SYSGEN> CONNECT CONSOLE
```

The commands in the following example autoconfigure the devices physically attached to the system and explicitly connect the network software device and the console block storage device:

```
SYSGEN>  AUTOCONFIGURE ALL
SYSGEN>  CONNECT NET/NOADAPTER/DRIVER=NETDRIVER
SYSGEN>  CONNECT CONSOLE
SYSGEN>  EXIT
```

Normally, the SYSGEN commands for connecting devices and loading device drivers are included in the site-independent startup command procedure (see Chapter 2).

Note that there is a DIGITAL-supplied driver named CONINTERR (SYS$SYSTEM:CONINTERR.EXE), which permits real-time processes to connect to interrupt vectors for quick response to and special handling of real-time events. The driver is not associated with any one device type. See the *VAX/VMS I/O User's Reference Manual: Part I* for further information.

11.9 Modifying System Files

When your system is installed or upgraded, AUTOGEN defines sizes for the primary paging, swapping, and dump files appropriate for your hardware configuration. The full file specification of each file is SYS$SYSTEM:filename.type. The filenames are PAGEFILE.SYS for the paging file, SWAPFILE.SYS for the swapping file, and SYSDUMP.DMP for the dump file. Sizes are expressed in pages.

AUTOGEN creates system files suitable for most systems. For special workloads or variant configurations, however, you must specify the file sizes with the SYSGEN command CREATE. (Or—to create primary files only—you can invoke a DIGITAL command procedure called SYS$UPDATE:SWAPFILES.COM and enter file sizes when the procedure prompts you for them.) The following example illustrates the use of the CREATE command:

```
SYSGEN>  CREATE PAGEFILE.SYS /SIZE=16384
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]PAGEFILE.SYS;1 extended
SYSGEN>  CREATE SWAPFILE.SYS /SIZE=7168
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SWAPFILE.SYS;1 extended
SYSGEN>  CREATE SYSDUMP.DMP /SIZE=2052
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SYSDUMP.DMP;1 extended
SYSGEN>  EXIT
```

The next example uses the command procedure:

```
$ @SYS$UPDATE:SWAPFILES
To leave a file size at its current value type a
carriage return in response to its size prompt.
Current file sizes are:
Directory SYS$SYSROOT:[SYSEXE]
PAGEFILE.SYS;1     16384
SYSDUMP.DMP;1       4128
SWAPFILE.SYS;1      3072
Total of 3 files, 23584 blocks.
There are 128741 available blocks on SYS$SYSDEVICE.
Enter new size for paging file: RET
Enter new size for system dump file: 6052
Enter new size for swapping file: RET
%SYSGEN-I-EXTENDED, SYS$SYSROOT:[SYSEXE]SYSDUMP.DMP;1 extended
************************************************************************
*  Please reboot in order for the new files to be used by the system.  *
*  After rebooting, purge obsolete copies of the files.                *
*  DO NOT delete the old files until after the reboot.                 *
************************************************************************
```

Both the command procedure and the CREATE command automatically extend the size of a paging or dump file if you specify a size that is larger than that of an existing file. (However, if you have explicitly installed a file with the SYSGEN command INSTALL, the file cannot be extended; instead, a new version of the file is created.) If you specify a smaller size for a system paging or dump file, a new version of the file is created. If the swapping file does not exist, you can create it using the SYSGEN command CREATE, or by responding to the appropriate prompt from the SWAPFILES procedure.

If you create a new version of a system file, you must delete the old version explicitly (but not until after the next bootstrap operation). In the case of a primary file (PAGEFILE.SYS, SWAPFILE.SYS, or SYSDUMP.DMP), the new or extended size file does not become effective until the system is shut down and restarted. In the case of a secondary file, the new file becomes effective when it is installed.

You can calculate appropriate sizes for your system files as follows:

- Paging file—Size of the average program at your site (in pages) times the maximum number of processes (MAXPROCESSCNT system parameter). The system

installation sets an initial size for your primary paging file. You can display statistics on your paging file usage with the DCL command SHOW MEMORY. Examine the data pertaining to the paging files. Aim to keep paging file usage less than half the size of the paging files. You can limit the usage of paging file space by user programs with the /PGFLQUOTA qualifier to the ADD and MODIFY commands in the Authorize Utility (see the *VAX/VMS Utilities Reference Volume*). You should not reduce the value of /PGFLQUOTA below 1024. Size requirements of the paging file can vary widely depending on user applications. Sufficient space in the paging file is critical to system performance. If a paging file starts to fill to the point where system performance is being affected, a message will be printed on the console terminal. Should this happen, you should increase the size of your paging file.

- Dump file—Size of physical memory in pages (to save the contents of memory if the system fails) plus four pages (to provide continuity of the error log when the system is shut down or if the system fails).
- Swapping file—Maximum number of processes (MAXPROCESSCNT system parameter) times the average of the working set quotas of processes running on the system. The system installation sets an initial size for SWAPFILE.SYS based on your hardware configuration. As an alternative to trying to calculate a more accurate swapping file size, you can monitor the swapping file usage with the DCL command SHOW MEMORY and watch its usage under load. You should aim to keep at least one-fourth of the swapping file space unused; otherwise system performance can be severely affected.

At bootstrap time, the system activates the latest versions of SYS$SYSTEM:PAGEFILE.SYS, SWAPFILE.SYS, and SYSDUMP.DMP. After bootstrapping, you can replace or augment the primary paging or swapping file with additional files (previously created with the SYSGEN command CREATE) by issuing the SYSGEN command INSTALL.

The advantages to using secondary files are that they do not have to be on the system disk and that they can span volumes in a volume set. Where secondary files are used, you should include SYSGEN INSTALL commands for the secondary files in the site-specific startup command procedure (see Chapter 2). If you are using secondary files because there

is a shortage of disk space on the system disk, you can reduce the size of PAGEFILE.SYS to a value as low as 4600 blocks, or SWAPFILE.SYS to a value as low as 1000 blocks.

All processes created after installation of additional paging files use the paging file with the most space, while all processes created before their installation continue to use the paging file to which they are assigned. Swapping space is allocated from whichever swapping file has space.

Installation of additional paging or swapping files requires nonpaged dynamic memory for a bit map, just as the primary files do.

11.10 Identifying a Startup Command Procedure for Bootstrap Operations

Chapter 2 describes the site-independent startup command procedure named SYS$SYSTEM:STARTUP.COM, which DIGITAL supplies and recommends that you do not edit. Following a bootstrap operation, the system executes the current site-independent startup command procedure. Initially (that is, as supplied in the software distribution kit), the procedure is STARTUP.COM.

However, if necessary, you can create alternate site-independent startup command procedures; for example, you can copy STARTUP.COM to other files of type COM and edit those files. You can then name an alternate site-independent command procedure to be invoked during subsequent bootstrap operations, using the SYSGEN command SET/STARTUP. You can use the SHOW/STARTUP command to display the current site-independent startup command procedure.

The SYSGEN commands in the following example display the current site-independent startup command procedure and specify an alternate:

```
SYSGEN>  SHOW/STARTUP
Startup command file = SYS$SYSTEM:STARTUP.COM
SYSGEN>  SET/STARTUP SYS$SYSTEM:XSTARTUP.COM
SYSGEN>  WRITE CURRENT
%OPCOM, 15-APR-1984 16:04:06.30, message from user SYSTEM
%SYSGEN-I-WRITECUR, CURRENT system parameters modified by process
ID 00160030 into file VAXVMSSYS.PAR
SYSGEN>  EXIT
```

11.11 Initializing Multiport (Shared) Memory Units

A single processor can attach one or two multiport memory units, each of which may vary in size from 256KB to 2MB. The front panel of each multiport memory unit displays the number of that unit. When you issue the SYSGEN command SHARE/INITIALIZE SYSGEN polls the processors with ports on the specified multiport memory unit. If no other processors are using the unit, the unit is initialized. If another processor is using the unit, the unit is connected. A SHARE/CONNECT command for an uninitialized multiport memory unit results in an error condition.

You should power down a multiport memory unit only after first shutting down and rebooting all systems connected to the unit.

A multiport memory unit managed by VAX/VMS contains the following preallocated structures, with space requirements as indicated:

- Common data page—Description of VAX/VMS data structure and quotas for the shared memory
- Global section description (GSD) table—Total global sections, as specified in the SHARE command, times 100 bytes, rounded up to the next full page
- Mailbox table—Total mailboxes, as specified in the SHARE command, times 48 bytes, rounded up to the next full page
- CEF table—Total common event flag clusters, as specified in the SHARE command, times 80 bytes, rounded up to the next full page
- PRQ pool—Total interprocess or request messages, as specified in the SHARE command, times 64 bytes, rounded up to the next full page
- Dynamic pool—Number of blocks allocated to the pool, as specified in the SHARE command, times the size of each block, as specified in the SHARE command
- Global section bit map—One page
- Global sections—Size of multiport memory unit minus the sum of the above preallocated structures (that is, the remaining space)

The following SYSGEN command SHARE specifies the default structure values for a 256KB multiport memory unit with four active ports. The resulting values are shown in Table 11–4.

```
SYSGEN>  SHARE MPMO SHR_MEM_1/INITIALIZE-
SYSGEN>  /GBLSECTIONS=32/MAILBOXES=32-
SYSGEN>  /CEFCLUSTERS=32/POOLBCOUNT=128-
SYSGEN>  /POOLBSIZE=128/PRQCOUNT=64
```

Table 11–4 Values for Multiport Memory Structures

Structure	Value	
CEF TABLE	32 x 80	= 5 pages
COMMON DATA PAGE	1	= 1 page
DYNAMIC POOL	128 x 128	= 32 pages
GLOBAL SECTIONS	512 - 57	= 455 pages
GLOBAL SECTION BIT MAP		= 1 page
GSD TABLE	32 x 100	= 7 pages
MAILBOX TABLE	32 x 48	= 3 pages
PRQ POOL	64 x 64	= 8 pages

The following guidelines are suggested in selecting values for the SHARE qualifiers that regulate the sizes of the preallocated structures:

- /CEFCLUSTERS, /GBLSECTIONS, and /MAILBOXES— You should simply specify the maximum number of each type of structure required by all processors at any one time. The same structure being used by many processes on one or more processors counts as just one structure.
- /POOLBCOUNT and /POOLBSIZE—The primary use of the dynamic pool is to buffer mailbox messages. The size of a message is 28 bytes plus the data in the message. Since space from the pool is always allocated in whole blocks, the recommended block size is the median message size plus 28. A block size that is too small for a message requires extra system overhead to concatenate the message blocks into the user buffer and segment them out of the user buffer. The number of blocks should be sufficient to satisfy all messages that might be outstanding at once. If a mailbox request cannot be satisfied due to insufficient pool space, the requesting process enters a resource wait state or the request fails (if resource wait mode is not enabled), just as if

the nonpaged dynamic pool were depleted. For this reason, you should tend to overestimate space requirements in the dynamic pool.

- /PRQCOUNT—The system uses interprocessor request blocks internally to transfer requests among the VAX/VMS executive routines and mailbox drivers on the different processors. PRQs are allocated and deallocated rapidly, so a large number should not be needed. The default value normally suffices. If an executive or driver request cannot be satisfied because PRQs are depleted, the requesting routine waits until a PRQ becomes available.

You should calculate the space remaining for the global sections and determine whether it is sufficient. If the space is insufficient, you might reduce the size of the dynamic pool. However, this condition really suggests the need for a larger or additional multiport memory unit.

Where a multiport memory unit is a normal part of the system configuration, you should include the SYSGEN commands to initialize and connect it in the site-specific startup command procedure (see Chapter 2).

11.12 Loading and Starting the MSCP Server

The Mass Storage Control Protocol (MSCP) is used to communicate between a VAX host and a DIGITAL Storage Architecture (DSA) controller. The MSCP server enables a VAX processor to make locally connected MASSBUS and UNIBUS disks available to all other users of a VAXcluster. (As system manager, however, you must decide which disks should be that widely available.)

You use the SYSGEN command MSCP to load and start the MSCP server. You then specify the served devices using the DCL command SET DEVICE/SERVED. For more information on the MSCP server and its functions, refer to the *Guide to VAXclusters*.

SYSGEN Parameters Used in AUTOGEN Calculations

The following tables list the SYSGEN parameters whose values AUTOGEN collects and (re)calculates in various operation phases.

Table 11–5 Parameters Collected in OLDSITE1.DAT for Use in Calculating New Values

GBLPAGES	GBLPAGFIL	GBLSECTIONS
MAXBUF	MAXPROCESSCNT	VIRTUALPAGECNT

Table 11–6 Parameters Collected in OLDSITE2.DAT and Propagated to New System if Old Current Value Exceeds Old Default Value

ACP_EXTCACHE	ACP_EXTLIMIT	ACP_FIDCACHE
CLISYMTBL	DEFMBXBUFQUO	DEFMBXMXMSG
DEFMBXNUMMSG	LRPSIZE	PQL_DASTLM
PQL_BDIOLM	PQL_DBYTLM	PQL_DDIOLM
PQL_DFILLM	PQL_DWSDEFAULT	PQL_DWSEXTENT
PQL_DWSQUOTA	PQL_MASTLM	PQL_MBIOLM
PQL_MBYTLM	PQL_MDIOLM	PQL_MFILLM
PQL_MWSDEFAULT	PQL_MWSEXTENT	PQL_MWSQUOTA
PROCSECTCNT	SRPSIZE	

Table 11–7 Parameters Collected in OLDSITE3.DAT and Propagated to New System if Old Current Value Differs from Old Default Value

ACP_BASEPRIO	ACP_DATACHECK	ACP_SWAPFLGS
ACP_WRITEBACK	BUGCHECKFATAL	BUGREBOOT
CRDENABLE	DEADLOCK_WAIT	DEFPRI
DISMOUMSG	DUMPBUG	EXTRACPU
LAMAPREGS	MAXSYSGROUP	MOUNTMSG
MVTIMEOUT	PAGFILCNT	PQL_DCPULM
PQL_DPRCLM	PQL_DTQELM	PQL_MCPULM
PQL_MPRCLM	PQL_MTQELM	REALTIME_SPTS
SAVEDUMP	SCSSYSTEMID	SWPFILCNT
TIMEPROMPTWAIT	TTY_ALTALARM	TTY_ALTYPAHD
TTY_BUF	TTY_DIALTYPE	TTY_OWNER
TTY_PARITY	TTY_PROT	TTY_RSPEED
TTY_SCANDELTA	TTY_SILOTIME	TTY_SPEED
TTY_TYPAHDSZ	UAFALTERNATE	USER3
USER4	USERD1	USERD2
XFMAXRATE		

Table 11–8 Version 4.0 Parameters Collected in OLDSITE4.DAT and Propagated to New System if Old Current Value Differs from Old Default Value

ALLOCLASS	DISK_QUORUM	LNMPHASHTBL
LNMSHASHTBL	LOCKDIRWT	PRCPOLINTERVAL
QDSKINTERVAL	QDSKVOTES	QUORUM
RECNXINTERVAL	SCSNODE	SCSSYSTEMIDH
TAILORED	TTY_DEFCHAR	TTY_DEFCHAR2
VAXCLUSTER	VOTES	

Table 11–9 Parameters Modified in AUTOGEN Calculations

ACP_DIRCACHE	ACP_HDRCACHE	ACP_MAPCACHE
ACP_MULTIPLE	ACP_QUOCACHE	ACP_SWAPFLGS
ACP_SYSACC	BALSETCNT	BORROWLIM
FREEGOAL	FREELIM	GBLPAGES
GBLPAGFIL	GBLSECTIONS	GROWLIM
IRPCOUNT	IRPCOUNTV	LOCKIDTBL
LOCKIDTBL_MAX	LRPCOUNT	LRPCOUNTV
LRPSIZE	MAXPROCESSCNT	MPW_HILIMIT
MPW_LOLIMIT	MPW_WAITLIMIT	NPAGEDYN
NPAGEVIR	PAGEDYN	PFCDEFAULT
PFRATL	RESHASHTBL	SPTREQ
SRPCOUNT	SRPCOUNTV	SRPSIZE
SYSMWCNT	VIRTUALPAGECNT	WSMAX

A Using BOOT58 on VAX–11/750 Systems

This appendix explains how to bootstrap VAX–11/750 systems using the stand-alone BOOT58 program stored on the TU58 tape cartridge.

In most cases it is quicker, easier, and recommended procedure to bootstrap the system directly from the system disk. When you do, the bootblock program must read and execute the boot block (block 0). If this block is corrupted, and the rest of the disk is intact, the only way to boot a VAX–11/750 system safely is to use stand-alone BOOT58. Later, you can use the Writeboot Utility (WRITEBOOT), described in Section A.2, to replace the bad boot block.

The stand-alone BOOT58 program resides on the console TU58 tape cartridge and is used to supplement the console subsystem's bootstrapping functions. When you request a boot from the console TU58 tape cartridge (device DDA0), the console subsystem microcode reads the TU58 boot block (block 0) into memory and executes it. The effect of executing the TU58 boot block is to load and transfer control to stand-alone BOOT58.

Using stand-alone BOOT58, you can boot the system either in a conversational manner or nonstop. In a conversational boot, you are allowed to intervene with system parameter changes as the boot is progressing. The nonstop boot does not allow you this flexibility.

The console TU58 tape cartridge contains two sets of bootstrap command procedures—a set of conversational procedures, and a set of nonstop procedures. To determine which procedures are available on your console volume, refer to Section 2.4.

Table A–1 summarizes BOOT58 commands.

Table A–1 BOOT58 Commands

Command	Function
BOOT [device-name]	Bootstraps the system from the specified device. If you omit the device name, the system is bootstrapped using the default bootstrap command procedure (DEFBOO.CMD). Note that you cannot specify this command within a command procedure, and that you cannot use the name of a command procedure as a command qualifier.
DEPOSIT [loc-qual, size-qual] location value	Deposits a value in the specified location. The location is interpreted according to the location and size qualifiers. The location qualifier can be expressed as follows: /G general register /I internal processor register /P physical memory The size qualifier can be expressed as follows: /B byte /W word /L longword If you do not specify the location and size qualifiers, the default values established by a previous command are used.

Table A–1 (Cont.) BOOT58 Commands

Command	Function
EXAMINE [loc-qual, size-qual] location	Displays the contents of the specified location. The location is interpreted according to the location and size qualifiers. The location qualifier can be expressed as follows: /G general register /I internal processor register /P physical memory The size qualifier can be expressed as follows: /B byte /W word /L longword If you do not specify the location and size qualifiers, the default values established by a previous command are used.
HELP	Displays the BOOT58 help file at the console terminal. You cannot specify this command within a command procedure.
LOAD file-spec [/START:address]	Loads a file from the bootstrap device into memory, starting at the address specified with the /START qualifier. If you omit the /START qualifier, the file is loaded into memory beginning at the first free address.

Table A–1 (Cont.) BOOT58 Commands

Command	Function
START value	Transfers control to the value specified. You generally use this command with the LOAD command.
@file-spec	Executes the name of the command procedure specified. You cannot specify a command procedure file-spec of more than six characters, nor can you specify nested command procedures.

A.1 Bootstrap Procedures

You can select one of the following procedures to bootstrap your VAX–11/750 system using BOOT58:

- Default
- Conversational
- Nonstop

These procedures are described in the next sections.

A.1.1 Default Bootstrap

To bootstrap the system using the default command procedure that you copied to the file DEFBOO.CMD, follow these steps:

1 Shut down the system by executing the SHUTDOWN command procedure:

```
$ @SYS$SYSTEM:SHUTDOWN
```

The command procedure prompts for the number of minutes until system shutdown, the reason for the shutdown, and whether to spin down the disks.

2 In response to the statement, "SYSTEM SHUTDOWN COMPLETE - USE CONSOLE TO HALT THE SYSTEM," halt the processor by pressing CTRL/P.

3 In response to the console prompt (>>>), enter the following command:

```
>>> B
```

4 When the BOOT58> prompt appears, enter the following command:

```
BOOT58> BOOT
```

A.1.2 Conversational Bootstrap

To do a conversational bootstrap using the stand-alone BOOT58 program, proceed as follows:

1 Insert the console TU58 tape cartridge (part description: BE-T204F-ME TU58#1 VAX–11/750 Local Console) into the console drive.

2 See that the rotary key switch on the processor control panel is set to LOCAL.

3 Set the POWER-ON-ACTION switch to HALT.

4 Press CTRL/P to obtain the console program prompt (>>>).

5 At the prompt, enter the following console BOOT command to boot the TU58 tape cartridge:

```
>>> B   DDA0
```

6 In response to the BOOT58> prompt, enter the name of a conversational bootstrap command procedure:

```
BOOT58> @DxyGEN
```

@ Indicates that the rest of the line contains the name of a command procedure located on the console TU58 tape cartridge.

x Indicates the device type of the desired bootstrap volume:

B—RM03, RM80, RP05, RP06, RP07
L—RL02
M—RK07
U—UDA

y Indicates the unit number of the desired bootstrap device volume. Specify a number in the range of 0 through 7.

When SYSBOOT is ready to accept commands, it prompts as follows:

```
SYSBOOT>
```

You can now issue any of the commands listed in Table 4-1.

If your system will not boot, you can enter the following sequence of SYSBOOT commands after the SYSBOOT> prompt to boot using default system parameter values:

```
SYSBOOT> USE DEFAULT
SYSBOOT> CONTINUE
```

The following example shows a conversational bootstrap procedure on a VAX–11/750 system using the stand-alone BOOT58 program. During the procedure, the user sets the value of the system parameter WSMAX to 512. Note that before displaying the BOOT58> prompt, the VAX–11/750 responds with a double percent sign to indicate successful verification of the microcode.

```
>>>B DDAO
%%
BOOT58> @DMOGEN
!
! DMO CONVERSATIONAL BOOT COMMAND FILE - DMOGEN
! BOOT FROM DMO AND STOP IN SYSBOOT TO ALTER PARAMETER VALUES
!
D/G 0 1                     ! CARTRIDGE DISK
D/G 1 FFE000                ! BASE OF UNIBUS I/O PAGE
D/G 2 3FF20                 ! CSR ADDRESS OFFSET
D/G 3 0                     ! CONTROLLER UNIT = 0
D/G 4 0                     ! BOOT BLOCK LBN    (UNUSED)
D/G 5 1                     ! SOFTWARE BOOT FLAGS  (CONVERSATIONAL)
D/G E 200                   ! ADDRESS OF WORKING MEMORY + ^X200
LOAD VMB.EXE/START:200      ! LOAD PRIMARY BOOTSTRAP
START 200                   ! START PRIMARY BOOTSTRAP
SYSBOOT> SET WSMAX 512
SYSBOOT> CONTINUE
  VAX/VMS Version 4.0 15-APR-1984-14:20:50.15

%%%%%%%%%%% OPCOM, 15-APR-1984-14:20:50.15 %%%%%%%%%%%
Logfile has been initialized by operator _OPAO:
Logfile is SYS$SYSROOT:[SYSMGR]OPERATOR.LOG;1
%SET-I-INTSET, login interactive limit = 64, Current interactive value = 0
  SYSTEM       job terminated at 15-APR-1984-14:20:50.15
```

A.1.3 Nonstop Bootstrap

If you want to bootstrap the system without stopping to change parameters, follow steps 1 through 5 for the conversational bootstrap procedure described in the previous section. Then, in response to the BOOT58> prompt, enter the name of a nonstop bootstrap command procedure in the following format:

```
BOOT58> @DxyBOO.CMD
```

@ Indicates that the rest of the line contains the name of a command procedure located on the console TU58 cartridge.

x Indicates the device type of the desired bootstrap device volume:

B—RM03, RM80, or RP06
M—RK07

y Indicates the unit number of the desired bootstrap volume. Specify a number in the range of 0 through 7.

If you use this form (that is, @DxyBOO.CMD), the contents of the command procedure are displayed on the console terminal.

The following example shows a nonstop bootstrap procedure on a VAX–11/750 system using the stand-alone BOOT58 program. Note that before displaying the BOOT58> prompt, the VAX–11/750 responds with a double percent sign to indicate successful verification of the microcode.

```
>>>B DDA0
%%
BOOT58> @DM0BOO.CMD

!
!      DM0 BOOT COMMAND FILE - DM0BOO.CMD
!
D/G 0 1                  ! CARTRIDGE DISK
D/G 1 FFE000             ! BASE OF UNIBUS I/O PAGE
D/G 2 3FF20              ! CSR ADDRESS OFFSET
D/G 3 0                  ! CONTROLLER UNIT = 0
D/G 4 0                  ! BOOT BLOCK LBN   (UNUSED)
D/G 5 0                  ! SOFTWARE BOOT FLAGS
D/G E 200                ! ADDRESS OF WORKING MEMORY + ^X200
LOAD VMB.EXE/START:200 ! LOAD PRIMARY BOOTSTRAP
START 200                ! START PRIMARY BOOTSTRAP
  VAX/VMS Version 4.0 15-APR-1984-14:20:50.15

%%%%%%%%%%% OPCOM, 15-APR-1984-14:20:50.15 %%%%%%%%%%%
Logfile has been initialized by operator _OPA0:
Logfile is SYS$SYSROOT:[SYSMGR]OPERATOR.LOG;1

%SET-I-INTSET, login interactive limit = 64, Current interactive value = 0
  SYSTEM       job terminated at 15-APR-1984-14:20:50.15
```

A.2 The Writeboot Utility

This section discusses the Writeboot Utility (WRITEBOOT), which writes a boot block on a bootable disk.

The normal bootstrap operation on the VAX–11/750 is done from the boot block (block 0) of the system disk. If this block is corrupted, the system can only be bootstrapped from the TU58. Once the system is running again, the bad boot block can be rewritten by using WRITEBOOT.

Note: The LOG_I/O privilege is required to use WRITEBOOT.

To invoke WRITEBOOT, type the following command at the DCL prompt:

```
$ RUN SYS$SYSTEM:WRITEBOOT
```

WRITEBOOT will prompt for input. Prompts and recommended responses follow.

```
Target system device (and boot file if not VMB.EXE):?
```

Enter the name of the device on which you want to rewrite the boot block (DMA0 for example).

Note: The boot file must be on the target disk.

The target system device should be specified like any VAX/VMS disk device (ddcu):

dd is the code name for the device type
c is the controller designation
u is the unit number

For example, DMA0 would be an RK07 on RK611 controller A, device number 0. If you want to use a boot file other than SYS$SYSTEM:VMB.EXE, you must provide the full file specification, including device and directory.

```
Enter VBN of boot file code (default is one):
```

If you are rewriting the boot block, press RETURN.

```
Enter load address of primary bootstrap in HEX (default is 200):
```

Again, if you are rewriting the boot block, press RETURN.

WRITEBOOT will then write the boot file that you specified in the first line, to the address that you entered in the third line.

The Writeboot Utility may issue one or more of the following error messages:

You lack LOG_IO privilege

Explanation: This message means you do not have the correct privilege to use WRITEBOOT.

You lack READ and/or WRITE access to TARGET DEVICE. DISMOUNT and REMOUNT

Explanation: This message means that access to the target device is limited. Check the **WRITE PROTECT** button.

VBN must be >= 1

Explanation: This message means you cannot specify a zero as the virtual block number (VBN).

B Technical Note on VAX/VMS CI

This appendix describes the computer interconnect (CI) and System Communication Services (SCS) including information on trouble-shooting and repairs. Information on CI entries in the system error log and on corrective actions to take when errors occur is also provided.

B.1 CI Port and System Communications Services Operation

The following sections describe the CI port and the operation of system communication services (SCS).

B.1.1 CI Port Polling and Virtual Circuits

Shortly after VAX/VMS boots, the CI port driver begins configuration polling in order to discover other active ports on the CI. Normally the poller runs every 5 seconds (the default value of SYSGEN parameter PAPOLLINT.) In the first polling pass, port numbers 0 through 15 are probed over cable path A; on the second pass port numbers 0 through 15 are probed over path B; on the third pass path A is probed again, and so on.

The poller probes by sending request id (REQID) packets to all possible port numbers, including itself. Active ports receiving the REQIDs return id packets (IDREC) to the port issuing the REQID. A port may respond to a REQID whether or not the system attached to the port is running.

Once a pair of ports and port drivers has successfully exchanged id packets, the port drivers perform a start handshake. They exchange datagrams containing information about the systems such as the type of CPU (for example, V780, V750, HSC) and the operating system version. If this exchange is successful, then each system declares a virtual circuit (VC) open. An open VC is prerequisite to all other activity. Each port supports up to 16 VCs.

B.1.2 System Communications Services Connections

System services such as the disk class driver, the VAXcluster connection manager, and the MSCP disk server communicate between systems with a protocol called System Communications Services (SCS). Primarily, SCS is responsible for the formation and breaking of intersystem process connections and for flow control of message traffic over those connections. In VAX/VMS Version 4.0, SCS is implemented in the CI port driver and in a loadable piece of the system called SCSLOA.EXE (loaded automatically during initialization).

When a VC has been opened, a VAX/VMS system periodically probes a remote system for system services that it may be offering. The SCS directory service, which makes known other services that its system is offering, is always present on both VAX/VMS and HSC systems. As system services discover their counterparts on other systems, they establish SCS connections to each other. These connections are full duplex and are associated with a particular VC. Multiple connections are typically associated with a virtual circuit.

B.1.3 Failures

Taken together, SCS, the CI port driver, and the CI port itself support a hierarchy of communications paths. Working up from the most fundamental level there are:

- The physical wires (two pairs of transmit and receive.) Path A transmit and receive and Path B transmit and receive. Normally, traffic is sent by VAX/VMS in automatic path select mode and the port chooses the free path or, if both are free, an arbitrary path (implemented in the cables, star coupler, and managed by the port).
- The port-to-port virtual circuit (implemented partly in the CI port and partly in system software).
- The connections (implemented in system software).

Failures can occur at each communications level and in each component. Failures at one level translate into failures at lower levels as follows:

- Wires. One of Path A or B can fail and the VC remains intact. All traffic is directed over the remaining good path.

If the wire is repaired, the repair is detected automatically by port polling, and done to all ports.

- Virtual circuit. If neither Path A nor B works between a pair of ports, then the VC fails and is closed. Path failures are discovered either by

  — Attempting to send normal traffic with the port reporting that neither path yielded transmit success

  — The poller sending REQIDs over Path A and B periodically when both paths are nonfunctional

  When a whole VC fails, every SCS connection on it fails. The software automatically reestablishes connections when the VC is reestablished. Normally reopening a VC takes about 10 seconds after the source of the problem disappears.

- Port. If a port fails, then all the VCs to that port fail and all SCS connections on those VCs fail. If the port is successfully reinitialized, VCs and connections will be reestablished automatically. Normally, reinitialization and connection reestablishment takes about 10 seconds.

- Connection. When the software protocols fail in some way or, in some instances, when the software thinks it detects a hardware malfunction, a connection is disconnected. Other connections are normally unaffected, as is the VC.

- System. If a whole system fails by operator shutdown, bugcheck, or halt and reboot, then its port fails and all other systems on the CI note this as failures of their VCs to the port on this system.

One interesting situation is the case where a system halts, but no INIT (11/750) or UNJAM (11/780) is done. The port remains running and other systems on the CI cannot detect the failed system. The cluster services on the failed system appear simply to be hung, because the port is an independent processor and continues to handle incoming traffic and to respond to REQIDs. (The same effect is achieved if the failed system is hung at IPL 8 or above.) VAX CI ports normally run with a sanity timer enabled, which puts the port into an uninitialized state detectable as failed VCs by other nodes if the host appears to be hung for 100 seconds.

B.2 Trouble-Shooting

This section contains advice on how to determine which system in the cluster and which component in that system is causing errors. These determinations can often be made without interrupting timesharing activity on systems in the cluster.

B.2.1 Adding Nodes to a Cluster

Before you bring up as a part of a cluster a system that is new, just repaired, or suspected of having a problem, you should verify that the system runs correctly on its own. Although this precaution cannot guarantee that cluster software will run correctly over the CI, it can help track down errors if they should occur.

If only an HSC disk is available for booting, you will be unable to verify the system on its own and you can proceed to step 2. If a local MASSBUS or UNIBUS disk is available for booting, you can use step 1 to boot your system almost entirely in isolation from CI activity.

1. Perform a conversational boot. While running SYSBOOT, suppress port activity with the following commands:

   ```
   SYSBOOT> SET PANOPOLL 1
   SYSBOOT> SET VAXCLUSTER 0
   ```

   Having booted in this way, you can run any desired tests to verify the system, independent of the CI.

   Also, since the above SYSGEN parameters still permit the CI port to be autoconfigured and initialized, you can verify minimal CI port capabilities. Use the following command to check that the port is initialized and is online:

   ```
   $ SHOW DEVICE PAAO
   ```

   The device should be online and have zero errors. If it has errors and/or is offline, check the error log. If the port failed to initialize because of device errors, please read the section of the *VAX/VMS Release Notes* concerning CI port microcode revision levels. Corrupted port microcode files and incompatible microcode file and proms are two reasons for continuous port errors upon initialization.

2 When the system has been verified (or if no local boot device is available), boot the system in a cluster from its normal boot device. Remember to do a conversational boot the first time and specify a unique SCSSYSTEMID and SCSNODE for the system. Also, if necessary, restore PANOPOLL to its default value of 0 and set VAXCLUSTER =1 (if the system will be joining a VAXcluster).

   As the system boots, watch the OPA0 terminal for %CNXMAN messages that indicate that this system is discovering other systems and forming connections to their connection managers. (See the *VAX/VMS Guide to Clusters,* Appendix B, for detailed descriptions of these messages.) Also, note any CI port error messages on OPA0. These are documented in section B.7 of this appendix.

   If the system fails to come all the way up so that you can log in and use it, but no obvious error information is on device OPA0, then check the status of the failing system from other VMS nodes using the SHOW CLUSTER/CONTINUOUS command of the Show Cluster Utility. See *VAX/VMS Utilities Reference Volume* for more detail.

3 In diagnosing these types of problem, it is useful to tailor the SHOW CLUSTER report by the command ADD CIRCUIT CABLE_ST. This command adds a class of information about all the virtual circuits as seen from the system on which you are running SHOW CLUSTER. Primarily, you are checking whether there is a virtual circuit in the OPEN state to the failing system. Common causes of failure to open a VC and keep it open are:

   - Port errors on one side or the other
   - Some cabling errors
   - A port set offline due to software problems
   - Insufficient nonpaged pool available on both sides
   - Failure to set the SYSGEN parameters SCSNODE, SCSSYSTEMID, PAMAXPORT, PANOPOLL, PASTIMOUT, and PAPOLLINT correctly

4 If no VC is open to the failing system, check the bottom of the SHOW CLUSTER display for circuit information to the port of the failing system. VCs in partially open states are placed at the bottom of the display. If the circuit is there in a state other than OPEN, communications between the

local port and remote port are taking place and the failure is probably at a higher level in the communications than in port or cable hardware. Secondarily, check that both Paths A and B are good to the failing port, although the loss of one path should not prevent a system from participating in a VAXcluster.

5 Run SHOW CLUSTER from each running system in the cluster to see that each system's view of the failing system is consistent with every other system's view. If all the running systems have a consistent view of the failing system, chances are the problem is in the failing system. If, on the other hand, only one of several running systems thinks the newcomer is failing, then perhaps the problem is in that one running system.

B.2.2 Crossed Cables and Loopback Datagrams

Whenever the configuration poller finds that no port-to-port virtual circuits are open and that no hand-shake procedures are currently opening virtual circuits, the poller analyzes its environment. It does so by using the send-loopback-datagram facility of the CI port.

The send-loopback-datagram facility tests the connections between the CI port and the Star coupler by routing a message across them. The messages are called loopback datagrams. (The port processes other self-directed messages without using the Star coupler or external cables.)

The configuration poller makes entries in the error log whenever it detects a change in the state of a circuit. Note, however, that it is possible for two changed-to-failed-state messages to be entered in the log without an intervening changed-to-succeeded-state message. Such a series of entries means that the circuit state continues to be faulty.

The following paragraphs discuss various incorrect CI cabling configurations and the entries made in the error log when these configurations exist. Figure B–1 shows a two-node configuration with all cables connected correctly.

Figure B–1 A Two-Node CI Cluster, Correctly Connected

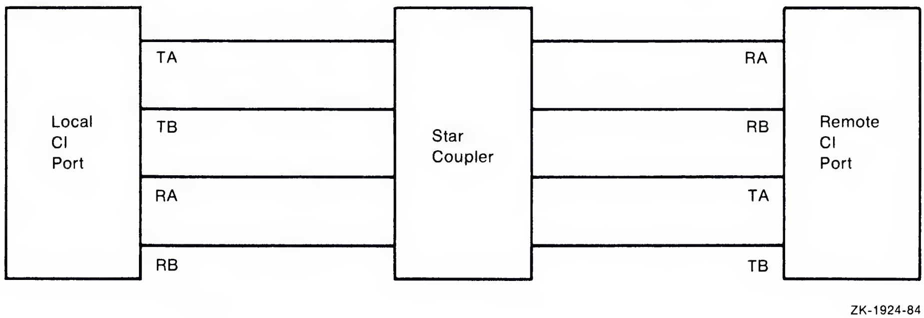
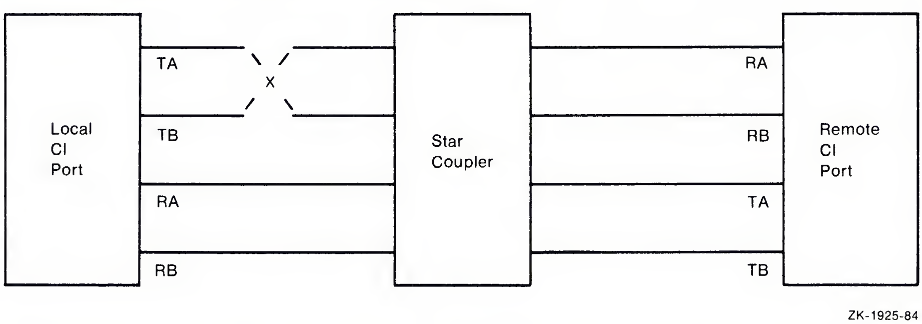


Figure B–2 shows a CI cluster with a pair of crossed cables. If a pair of transmitting cables or a pair of receiving cables is crossed, a message sent on TA is received on RB, and a message sent on TB is received on RA. This is a hardware error condition from which the port cannot recover. An entry is made in the error log to say that a single pair of crossed cables exists. The entry contains the lines

```
DATA CABLE(S) CHANGE OF STATE
PATH  1.  LOOPBACK HAS GONE FROM GOOD TO BAD
```

Figure B–2 Crossed CI Cable Pair



If this situation exists, you can correct it by reconnecting the cables properly. The cables could be misconnected in several places. The coaxial cables that connect the port boards to the bulkhead cable connectors can be crossed, or the Ethernet cables can be misconnected to the bulkhead or the STAR coupler.

The information in Figure B–2 can be represented more simply.

The example below shows the cables positioned as in Figure B–2, but it does not show the star coupler or the nodes. The letters LOC and REM indicate the pairs of transmitting (T) and receiving (R) cables on the local and remote nodes, respectively.

```
T x   = R
R =   = T
LOC   REM
```

The pair of crossed cables causes loopback datagrams to fail on the local node, but succeed on the remote node. Crossed pairs of transmitting cables and crossed pairs of receiving cables cause the same behavior.

Note that only an odd number of crossed-cable pairs causes these problems. If an even number of cable pairs is crossed, communications succeed. An error-log entry is made in some cases, however, and the contents of the entry depends on which pairs of cables are crossed.

The second example shows two-node clusters with the combinations of two pairs of crossed-cable pairs.

```
T x   = R       T =   x R
R x   = T       R =   x T
LOC   REM       LOC   REM
```

These crossed pairs cause the following entry to be made in the error log of the node that has the cables crossed:

```
DATA CABLE(S) CHANGE OF STATE
CABLES HAVE GONE FROM UNCROSSED TO CROSSED
```

Loopback datagrams succeed on both nodes, and communications are possible.

The third example shows the possible combinations of two pairs of crossed cables that cause loopback datagrams to fail on both nodes in the cluster. Communications can take place between the nodes nevertheless. An entry stating that cables are crossed is made in the error log of each node.

```
T x   = R       T =   x R
R =   x T       R x   = T
LOC   REM       LOC   REM
```

The fourth example shows the possible combinations of two pairs of crossed cables that cause loopback datagrams to fail on both nodes in the cluster, but allow communications. No entry stating that cables are crossed is made in the error log of either node.

```
T x   x R       T =   = R
R =   = T       R x   x T
LOC   REM       LOC   REM
```

The fifth example shows the possible combinations of three pairs of crossed cables. In each case, loopback datagrams fail on the node that has only one crossed pair of cables. Loopback datagrams succeed on the node with both pairs crossed. No communications are possible.

```
T x   x R       T x   = R       T =   x R       T x   x R
R x   = T       R x   x T       R x   x T       R =   x T
LOC   REM       LOC   REM       LOC   REM       LOC   REM
```

If all four cable pairs between two nodes are crossed, communications succeed, loopback datagrams succeed, and no crossed-cable message entries are made in the error log. Such a condition might be detected by noting error-log entries made by a third system in the cluster, but only if the third node has one of the crossed-cable cases described above.

B.3 Repair of Cables

This section describes some ways in which DIGITAL Field Service can make repairs on a running system. This information is provided to aid system managers in scheduling repairs.

In order for the software to survive cable-checking activities or cable-replacement activities, you must be sure that at least Path A or Path B is intact at all times between each port and every other port in the cluster.

You can, for example, remove Path A and Path B in turn from a particular port to the star coupler. To make sure that the configuration poller finds a path that was previously faulty but is now without fault, you must time your operations as follows:

1. Remove Path B.
2. After the poller has discovered that Path B is faulty, reconnect Path B.
3. Wait two poller intervals (or run SHOW CLUSTER) to make sure that the poller has reestablished Path B. Or, run SHOW CLUSTER/CONTINUOUS and ADD CIRCUITS, CABLE_ST; wait until SHOW CLUSTER tells you that Path B has been regained.
4. Remove Path A.
5. After the poller has discovered that Path A is faulty, reconnect Path A.
6. Wait two poller intervals to make sure that the poller has reestablished Path A.

If both paths are lost at the same time, then the port-to-port virtual circuits are lost between the port with the broken cables and all other ports in the cluster. This will in turn result in loss of SCS connections over the broken virtual circuits. However, recovery from this situation is automatic after an interruption in service on the affected node. The length of the interruption varies but is usually about two poller intervals or 10 seconds at the default SYSGEN parameter settings.

B.4 Analyzing Cluster Error-Log Entries

Part of anticipating and avoiding problems is to monitor the activities of the system; important events are recorded in the error log. From the total error count, displayed by the DCL command SHOW DEVICE PA, you can tell if errors have been increasing. If so, you should examine the error log.

The ANALYZE/ERROR_LOG command invokes the Error Log Utility to report the contents of an error-log file. For more information, see the *VAX/VMS Utilities Reference Volume.*

Note: **Some error-log entries are informational only, and require no action. If you shut down a system in the cluster, for example, all other systems in the cluster that have open port-to-port virtual circuits between themselves and the shutdown system make entries in their error logs. Such systems record up to three errors for the event: Path A got no response, Path B got no response, the virtual circuit is being closed. These messages are normal and reflect the change of state in the circuits to the shutdown system.**

On the other hand, some error-log entries are made for problems that degrade operation, or for nonfatal hardware problems. VAX/VMS might continue to run satisfactorily under these conditions. The purpose of detecting these problems early is to prevent nonfatal problems (such as loss of a single path) from becoming serious problems (such as loss of both paths).

B.5 Error-Log Entry Formats

Errors and other events on the CI cause the port driver to enter information into the system error log. The two formats used for CI error-log entries are the device-attention format and the logged-message format. This section describes those formats.

Device-attention entries record events that, in general, are indicated by the setting of a bit in a hardware register. Logged-message entries record the receipt of a message packet that contains erroneous data or that signals an error condition.

B.5.1 Device-Attention Entries

Example B–1 shows a device-attention entry. The left column gives the name of a device register or a memory location, the center column gives the value contained in that register or location, and the right column gives an interpretation of that value.

Example B–1 Device-Attention Entry

```
***************************** ENTRY     83. *****************************   1
ERROR SEQUENCE 10.                                    LOGGED ON SID 0150400A
DEVICE ATTENTION, 17-JUL-1984 17:11:33.99 KA780 REV #10. SERIAL #10.        2
CI SUB-SYSTEM, STAR$PAA0: - PORT POWER DOWN                                 3
      CNFGR            00800038
                                        ADAPTER IS CI
                                        ADAPTER POWER-DOWN
      PMCSR            000000CE
                                        MAINTENANCE TIMER DISABLE
                                        MAINTENANCE INTERRUPT ENABLE
                                        MAINTENANCE INTERRUPT FLAG
                                        PROGRAMMABLE STARTING ADDRESS
                                        UNINITIALIZED STATE
      PSR              80000001
                                        RESPONSE QUEUE AVAILABLE
                                        MAINTENANCE ERROR
      PFAR             00000000
      PESR             00000000
      PPR              03F80001
      UCB$B_ERTCNT           32                                             4
                                        50. RETRIES REMAINING
      UCB$B_ERTMAX           32                                             5
                                        50. RETRIES ALLOWABLE
      UCB$L_CHAR       0C450000
                                        SHAREABLE
                                        AVAILABLE
                                        error-logging
                                        CAPABLE OF INPUT
                                        CAPABLE OF OUTPUT
      UCB$W_STS            0010
                                        ONLINE
      UCB$W_ERRCNT         000B                                             6
                                        11. ERRORS THIS UNIT
```

❶ The first two lines are the entry heading. These lines contain the number of the entry in this error-log file, the sequence number of this error, and the identification number (SID) of this system's CPU. Each entry in the log file contains such a heading.

❷ The next line contains the entry type (DEVICE ATTENTION), the date, the processor type (KA780), the hardware revision number of the CPU (REV #10), and the serial number of the CPU (SERIAL #10).

❸ The line CI SUB-SYSTEM, STAR$PAA0: - PORT POWER DOWN contains the name of the subsystem and the device that caused the entry, and the reason for the entry. The CI subsystem's device PAA0 on node STAR was powered down.

The next 15 lines contain the names of hardware registers in the port, their contents, and interpretations of those contents. See the *CI780/CI750 User's Guide* for a description of all the CI port registers.

The CI port can recover from many errors, but not all. When an error occurs from which the CI cannot recover, the port notifies the port driver. The port driver logs the error and attempts to reinitialize the port. If the port fails after 50 such initialization attempts, the driver takes it offline unless the system disk is connected to the failing port or this system is supposed to be a cluster member. If the CI port is required for system disk access or cluster participation and all 50 reinitialization attempts have been used, then the system bugchecks with a CIPORT-type bugcheck. Once a CI port is offline, you can put the port back online only by rebooting the system.

❹ The UCB$B_ERTCNT field contains the number of reinitializations that the port driver can still attempt. The difference between this value and UCB$B_ERTMAX is the number of reinitializations already attempted.

❺ The UCB$B_ERTMAX field contains the maximum number of times the port can be reinitialized by the port driver.

❻ The UCB$W_ERRCNT field contains the total number of errors that have occurred on this port since it was booted. This total includes both errors that caused reinitialization of the port, and errors that did not.

B.5.2 Logged-Message Entries

Logged-message entries are made when the CI port receives a response that contains either data that the port driver cannot interpret, or an error code in the status field of the response.

Example B–2 shows a logged-message entry with an error code in the status field PPD$B_STATUS.

Example B–2 Logged-Message Entry

```
***************************** ENTRY     3. *****************************
ERROR SEQUENCE 3.                                  LOGGED ON SID 01188542
ERL$LOGMESSAGE, 17-JUL-1984 13:40:25.13 KA780 REV #3. SERIAL #1346.
CI SUB-SYSTEM, STAR$PAA0: - DATA CABLE(S) CHANGE OF STATE               ❶
      PATH #0. HAS GONE FROM GOOD TO BAD                                 ❷
      LOCAL STATION ADDRESS, 000000000002 (HEX)                          ❸
      LOCAL SYSTEM ID, 000000000001 (HEX)                                ❹
      REMOTE STATION ADDRESS, 000000000004 (HEX)                         ❺
      REMOTE SYSTEM ID, 0000000000A9 (HEX)                               ❻
      UCB$B_ERTCNT          32                                           ❼
                                  50. RETRIES REMAINING
      UCB$B_ERTMAX          32
                                  50. RETRIES ALLOWABLE
      UCB$W_ERRCNT        0001
                                  1. ERRORS THIS UNIT
      PPD$B_PORT            04                                           ❽
                                  REMOTE NODE #4.
      PPD$B_STATUS          A5                                           ❾
                                  FAIL
                                  PATH #0., NO RESPONSE
                                  PATH #1., "ACK" OR NOT USED
                                  NO PATH
      PPD$B_OPC             05                                           ❿
                                  IDREQ
      PPD$B_FLAGS           03                                           ⓫
                                  RESPONSE QUEUE BIT
                                  SELECT PATH #0.
      "CI" MESSAGE                                                       ⓬
                      00000000
                      00000000
                      80000004
                      0000FE15
                      4F503000
                      00000507
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
                      00000000
```

❶ The first line following the heading gives the reason for the entry: one or more data cables have changed state.

❷ The next line gives a more detailed reason for the entry: path 0, which the port used successfully before, cannot be used now.

Note: **ANALYZE/ERROR_LOG uses the notation path 0 and path 1; cable labels use the notation path A (=0) and path B (=1).**

The local ❸ and remote ❺ station addresses are the port numbers (range 0–15) of the local and remote ports. The port numbers are set in hardware switches by field service. The local ❹ and remote ❻ system ID are the SCS system IDs set by the SYSGEN parameter SCSSYSTEMID for the local and remote VAX systems. For HSCs, the system ID is set with the HSC console.

❼ The rest of the entry, which contains the entry fields that begin with UCB$, gives information on the contents of the unit control block (UCB) for this CI device.

The following fields, which begin with PPD$, are fields in the message packet that the local port has received.

❽ PPD$B_PORT contains the station address of the remote port. In a loopback datagram, however, this field contains the local station address.

❾ The PPD$B_STATUS field contains information on the nature of the failure that occurred during the current operation. When the operation completes without error, ERF prints the word NORMAL beside this field; otherwise, ERF decodes the error information contained in PPD$B_STATUS. Here a NO PATH error occurred because of a lack of response on path 0, the selected path.

❿ The PPD$B_OPC field contains the code for the operation that the port was attempting when the error occurred. The port was trying to send a request-for-id message.

⓫ The PPD$B_FLAGS field contains bits that indicate, among other things, the path that was selected for the operation.

⓬ The "CI" MESSAGE is a hexadecimal listing of bytes 16 through 83 (base 10) of the response (message or datagram). Since responses are of variable length depending upon the port opcode, bytes 16 through 83 may contain either more or fewer bytes than actually belong to the message. Here the request-for-id contains no information in bytes 16 through 83.

B.6 Error-Log Entry Descriptions

This section describes error-log entries. Each entry below is followed by a brief description of what the CI port driver does, and the suggested action a system manager should take. In cases where Software Performance Reports with crash dumps are requested, it is important to capture the crash dumps as soon as possible after the error. Note that path A and path 0 are the same path, and that path B and path 1 are the same path.

11/750 CPU MICROCODE NOT ADEQUATE FOR CI

The CI port driver: Sets the port offline with no retries attempted. In addition, if this port is needed because the system is booted from an HSC or is participating in a VAXcluster, the system bugchecks with a UCODEREV code bugcheck.

User Action: Read the *VAX/VMS Release Notes* and any subsequent *VAX/VMS Release Notes* for information on required CPU microcode revisions. Call Field Service if necessary.

CI PORT MICROCODE REV NOT CURRENT, BUT SUPPORTED

The CI port driver: Detected that the microcode is not at the current level, but will continue normally. This error is logged as a warning only.

User Action: Contact Field Service when convenient to have the microcode updated.

CI PORT MICROCODE REV NOT SUPPORTED

The CI port driver: Sets the port offline without attempting any retries.

User Action: Read the *VAX/VMS Release Notes* and any subsequent *VAX/VMS Release Notes* for information on the required CI port microcode revision. Contact Field Service if necessary.

DATA CABLE(S) CHANGE OF STATE
CABLES HAVE GONE FROM CROSSED TO UNCROSSED

The CI port driver: Logs this event.

User Action: No action needed.

DATA CABLE(S) CHANGE OF STATE
CABLES HAVE GONE FROM UNCROSSED TO CROSSED

The CI port driver: Logs this event.

User Action: Check for crossed-cable pairs. See Section B.2.

DATA CABLE(S) CHANGE OF STATE
PATH 0. HAS GONE FROM BAD TO GOOD

The CI port driver: Logs this event.

User Action: No action needed.

DATA CABLE(S) CHANGE OF STATE
PATH 0. HAS GONE FROM GOOD TO BAD

The CI port driver: Logs this event.

User Action: Check path A cables to see that they are not broken or improperly connected.

DATA CABLE(S) CHANGE OF STATE
PATH 0. LOOPBACK HAS BECOME GOOD, UNCROSSED

The CI port driver: Logs this event.

User Action: No action needed.

DATA CABLE(S) CHANGE OF STATE
PATH 0. LOOPBACK HAS GONE FROM GOOD TO BAD

The CI port driver: Logs this event.

User Action: Check for crossed-cable pairs or faulty CI hardware. See Section B.2, Trouble-Shooting.

DATA CABLE(S) CHANGE OF STATE
PATH 1. HAS GONE FROM BAD TO GOOD

The CI port driver: Logs this event.

User Action: No action needed.

DATA CABLE(S) CHANGE OF STATE
PATH 1. HAS GONE FROM GOOD TO BAD

The CI port driver: Logs this event.

User Action: Check path B cables to see that they are not broken or improperly connected.

DATA CABLE(S) CHANGE OF STATE
PATH 1. LOOPBACK HAS BECOME GOOD, UNCROSSED

The CI port driver: Logs this event.

User Action: No action needed.

DATA CABLE(S) CHANGE OF STATE
PATH 1. LOOPBACK HAS GONE FROM GOOD TO BAD

The CI port driver: Logs this event.

User Action: Check for crossed-cable pairs or faulty CI hardware. See Section B.2, Trouble-Shooting.

DATAGRAM FREE QUEUE INSERT FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

DATAGRAM FREE QUEUE REMOVE FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures, or memory, SBI (11/780), or CMI (11/750) contention.

FAILED TO LOCATE PORT MICRO-CODE IMAGE

The CI port driver: Marks device offline and makes no retries.

User Action: Make sure console volume contains the microcode file CI780.BIN, and then reboot the system.

HIGH PRIORITY COMMAND QUEUE INSERT FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

HSC ERROR-LOGGING DATAGRAM RECEIVED

The CI port driver: Upon receipt of an error message from the HSC, the port driver logs the error and takes no other action. It is recommended that you disable the sending of HSC informational error-log datagrams with the appropriate HSC console command. Informational error-log datagrams take considerable space in the error-log data file.

User Action: These are useful to read only if they are not captured on the HSC console for some reason (for example, the HSC console ran out of paper.) This logged information is a duplicate of the messages logged on the HSC console.

INAPPROPRIATE "SCA" CONTROL MESSAGE

The CI port driver: Closes the port-to-port virtual circuit to the remote port.

User Action: Submit a Software Performance Report to DIGITAL including the error logs and the crash dumps from the local and remote systems.

INSUFFICIENT NON-PAGED POOL FOR INITIALIZATION

The CI port driver: Marks device offline and makes no retries.

User Action: Reboot the system with a larger value for NPAGEDYN or NPAGEV.

LOW PRIORITY COMMAND QUEUE INSERT FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

MESSAGE FREE QUEUE INSERT FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

MESSAGE FREE QUEUE REMOVE FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

MICRO-CODE VERIFICATION ERROR

The CI port driver: Detected an error while reading the microcode that it just loaded into the port. The driver attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service.

NO PATH-BLOCK DURING "VIRTUAL CIRCUIT" CLOSE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Submit a Software Performance Report to DIGITAL including the error log and a crash dump from the local system.

NO TRANSITION FROM "UNINITIALIZED" TO "DISABLED"

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service.

PORT ERROR BIT(S) SET

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service.

PORT HAS CLOSED "VIRTUAL CIRCUIT"

The CI port driver: Closes the virtual circuit that the local CI port opened to the remote port.

User Action: Check the PPD$B_STATUS field of the error-log entry for the reason the virtual circuit was closed. This error is normal if the remote system crashed or was shut down.

PORT POWER DOWN

The CI port driver: Halts port operations, and then waits for power to return to the port hardware.

User Action: Restore power to the port hardware.

Note: **Powering the CI port down from the port's power supply box and then up while there is any traffic in transit through the port is not supported in VAX/VMS Version 4.0; however, CPU and whole system power failures are fully supported.**

PORT POWER UP

The CI port driver: Reinitializes the port and restarts port operations.

User Action: No action needed.

RECEIVED "CONNECT" WITHOUT PATH-BLOCK

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Submit a Software Performance Report to DIGITAL including the error log and a crash dump from the local system.

REMOTE SYSTEM CONFLICTS WITH KNOWN SYSTEM

The CI port driver: The configuration poller discovered a remote system with SCSSYSTEMID and/or SCSNODE equal to that of another system to which a virtual circuit is already open.

User Action: Shut the new system down as soon as possible. Reboot it with a unique SCSYSTEMID and SCSNODE. Do not leave the new system up any longer than necessary. If you are running a VAXcluster and two systems with conflicting identity are polling when any other virtual circuit failure takes place in the VAXcluster, then systems in the VAXcluster may crash with a CLUEXIT bugcheck.

Note: VAX/VMS Version 3.0 systems are exempt from the SCSNODE uniqueness requirement since they all have built-in node names of all blanks. Neither can Version 3.0 systems participate in Version 4.0 style VAXclusters. The purpose of the exemption is to allow customers to phase over from Version 3.0 to Version 4.0-style clusters.

RESPONSE QUEUE REMOVE FAILURE

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service. This error is caused by a failure to obtain access to an interlocked queue. Possible sources of the problem are CI hardware failures or memory, SBI (11/780), or CMI (11/750) contention.

SCSSYSTEMID MUST BE SET TO NONZERO VALUE

The CI port driver: Sets the port offline without attempting any retries.

User Action: Reboot the system with a conversational boot and set the SCSSYSTEMID to the correct value. At the same time, check that SCSNODE has been set to the correct nonblank value.

SOFTWARE IS CLOSING "VIRTUAL CIRCUIT"

The CI port driver: Closes the virtual circuit to the remote port.

User Action: Check error-log entries for the cause of the virtual circuit closure. Faulty transmission or reception on both paths, for example, causes this error and may be detected from the one or two previous error-log entries noting bad paths to this remote node.

SOFTWARE SHUTTING DOWN PORT

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Check other error-log entries for the possible cause of the port reinitialization failure.

UNEXPECTED INTERRUPT

The CI port driver: Attempts to reinitialize the port; after 50 failing attempts, it marks the device offline.

User Action: Call DIGITAL Field Service.

UNRECOGNIZED "SCA" PACKET

The CI port driver: Closes the port-to-port virtual circuit to the remote port. If the virtual circuit is already closed, the port driver inhibits datagram reception from the remote port.

User Action: Submit a Software Performance Report to DIGITAL, including the error-log file that contains this entry and the crash dumps from both the local and remote systems.

B.7 OPA0 Error Messages

There are a number of error conditions detected by the CI port driver (PADRIVER) which result in an attempt by the driver to log the error condition. Under a certain set of circumstances, the attempt to log the error to the error-logging device will fail. Such failures often happen because the error-logging device is not accessible, for one reason or another, at the time the attempt is made to log the error condition. Because of the central role the CI plays in VAXclusters, the loss of error-logged information in such cases makes it quite difficult to diagnose and fix problems.

A second, redundant method of error-logging captures at least some of the information about CI error conditions which would otherwise be lost. This second method consists of broadcasting selected information about the error condition to OPA0, in addition to the port driver's attempt to log the error condition to the error-logging device. The CI port driver attempts both OPA0 error broadcasting and standard error logging under any of the following circumstances:

- The system disk has not yet been mounted.
- The system disk is undergoing mount verification.
- During mount verification, it is discovered that the system disk drive contains the wrong volume.
- The system disk has timed out.
- The local system is participating in a cluster and quorum has been lost.

Note the implicit assumption that the system and error logging devices are one and the same.

This second method of reporting errors is also not entirely foolproof. Because of the manner in which the OPA0 error broadcasting must be performed, it is possible that some error conditions may not be reported. This situation will occur whenever a second error condition is detected before the CI port driver has had a chance to broadcast the first error condition to OPA0. In such a case, only the first error condition is reported to OPA0, since it is deemed to be the more important one.

Certain error conditions are always broadcast to OPA0, regardless of whether the error-logging device is accessible. In general, these are errors that cause the port to shut down either permanently or temporarily.

There is one OPA0 error message for each error condition also subjected to standard error-logging. The text of each error message is similar to the summary which would otherwise by displayed by formatting the corresponding standard error-log entry using Error Log Utility. See Section B.6 for a list of the Error Log Utility summary messages and their explanations.

Many of the OPA0 error messages contain some optional information such as the remote port number, CI packet information (flags, port operation code, response status, and port number fields), or specific CI port registers.

Following is a list of each of the OPA0 error messages, subdivided by error type. The *CI780/CI750 User's Guide* contain a detailed description of the CI port registers (CNF = Configuration Register; PMC = Port Maintenance and Control Register; PSR = Port Status Register), which are optionally displayed for certain of the error conditions. The codes, always finaccessible, specify whether the message is always logged on OPA0 or only when the system device is inaccessible.

Software Errors During Initialization (Always Logged on OPA0)

```
%PAx0, Insufficient Non-paged Pool for Initialization
%PAx0, Failed to Locate Port Micro-code Image
%PAx0, SCSSYSTEMID has NOT been set to a Non-zero Value
```

Hardware Error (Always Logged on OPA0)

%PAx0, Micro-code Verification Error

%PAx0, Port Transition Failure - CNF/PMC/PSR xxxxxxxx/xxxxxxxx/xxxxxxxx

%PAx0, Port Error Bit(s) Set - CNF/PMC/PSR xxxxxxxx/xxxxxxxx/xxxxxxxx

%PAx0, Port Power Down

%PAx0, Port Power Up

%PAx0, Unexpected Interrupt - CNF/PMC/PSR xxxxxxxx/xxxxxxxx/xxxxxxxx

%PAx0, CI port ucode not at required rev level. RAM/PROM rev is xxxx/xxxx

%PAx0, CI port ucode not at current rev level. RAM/PROM rev is xxxx/xxxx

%PAx0, CPU ucode not at required rev level for CI activity

Queue Interlock Failures (Always Logged on OPA0)

%PAx0, Message Free Queue Remove Failure

%PAx0, Datagram Free Queue Remove Failure

%PAx0, Response Queue Remove Failure

%PAx0, High Priority Command Queue Insert Failure

%PAx0, Low Priority Command Queue Insert Failure

%PAx0, Message Free Queue Insert Failure

%PAx0, Datagram Free Queue Insert Failure

Errors Signaled with a CI Packet

%PAx0, Unrecognized SCA Packet - FLAGS/OPC/STATUS/PORT xx/xx/xx/xx
(ALWAYS)

%PAx0, Port has Closed Virtual Circuit - REMOTE PORT xxx
(ALWAYS)

%PAx0, Software Shutting Down Port
(ALWAYS)

%PAx0, Software is Closing Virtual Circuit - REMOTE PORT xxx
(ALWAYS)

%PAx0, Received Connect Without Path-Block - FLAGS/OPC/STATUS/PORT xx/xx/xx/xx
(ALWAYS)

%PAx0, Inappropriate SCA Control Message - FLAGS/OPC/STATUS/PORT xx/xx/xx/xx
(ALWAYS)

%PAx0, No Path-Block During Virtual Circuit Close - REMOTE PORT xxx
(ALWAYS)

%PAx0, HSC error-logging Datagram Received - REMOTE PORT xxx
(INACCESSIBLE)

%PAx0, Remote System Conflicts with Known System - REMOTE PORT xxx
(ALWAYS)

Cable Change-of-State Notification

```
%PAx0, Path #0. Has gone from GOOD to BAD - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Path #1. Has gone from GOOD to BAD - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Path #0. Has gone from BAD to GOOD - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Path #1. Has gone from BAD to GOOD - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Cables have gone from UNCROSSED to CROSSED - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Cables have gone from CROSSED to UNCROSSED - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Path #0. Loopback has gone from GOOD to BAD - REMOTE PORT xxx
    (ALWAYS)
%PAx0, Path #1. Loopback has gone from GOOD to BAD - REMOTE PORT xxx
    (ALWAYS)
%PAx0, Path #0. Loopback has gone from BAD to GOOD - REMOTE PORT xxx
    (ALWAYS)
%PAx0, Path #1. Loopback has gone from BAD to GOOD - REMOTE PORT xxx
    (ALWAYS)
%PAx0, Path #0. Has become working but CROSSED to Path #1. - REMOTE PORT xxx
    (INACCESSIBLE)
%PAx0, Path #1. Has become working but CROSSED to Path #0. - REMOTE PORT xxx
    (INACCESSIBLE)
```

Two other messages concerning the CI port appear on OPA0. They are:

```
%PAx0, CI port is reinitializing (xxx retries left.)
%PAx0, CI port is going offline.
```

The first message indicates that a previous error requiring the port to shut down is recoverable and that the port will be reinitialized. The 'xxx retries left' is how many more reinitializations are allowed before the port must be left permanently offline. Each reinitialization of the port (for reasons other than power fail recovery) causes approximately 2 Kbytes of nonpaged pool to be lost.

The second message indicates that a previous error is not recoverable and the port will be left offline. The only way to recover the port in this case is to reboot the system.

Index

A

B

F

G

H

I

J

K

L

M

T

Guide to VAX/VMS
System Management
and Daily Operations
AA-Y507A-TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) ______

Name ______ Date ______

Organization ______

Street ______

City ______ State ______ Zip Code or Country ______

Do Not Tear - Fold Here and Tape

digital

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line